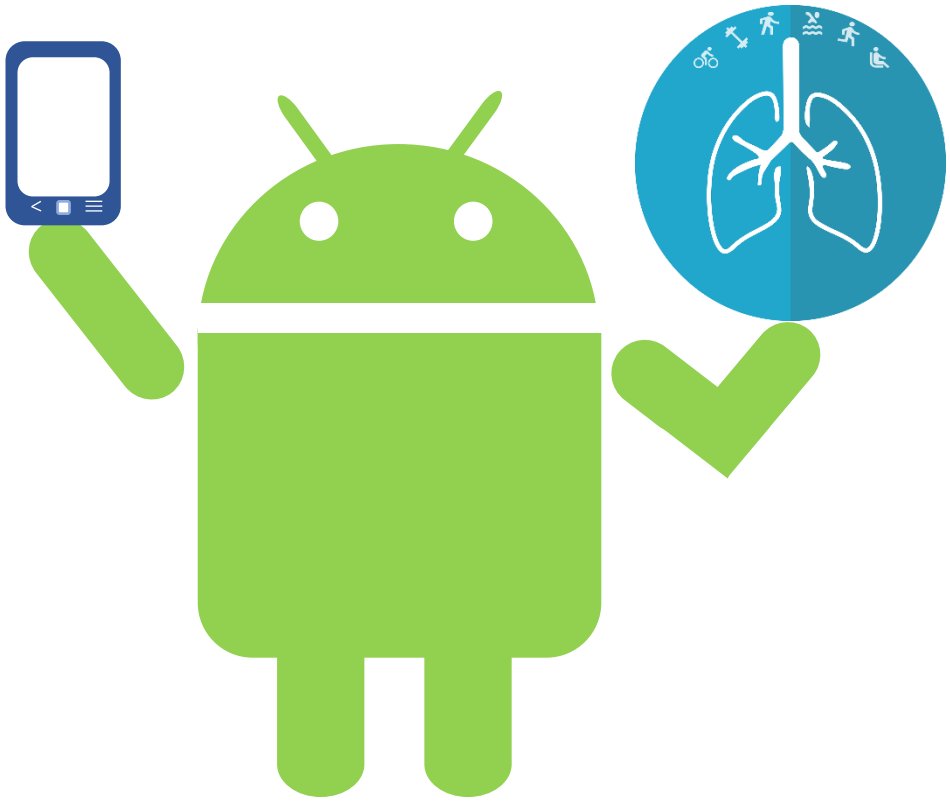
Universidad Carlos III de Madrid

Trabajo Fin de Grado



“Aplicación Android para medir la frecuencia respiratoria”

Grado Ingeniería Telemática

Autor: Sergio García Águila

Tutor Proyecto: Mario Muñoz Organero

ABSTRACT

From some years ago until now, we are living the digital transformation. One of the pillars of this digital transformation is the revolution of the smartphones. At the beginning, the mobile phone was intended for communication between people, but now, we use it for many other things. Pay for our purchases, post on our social networks, listen to music, maintain contact between companies and their customers, read the news, and so forth.

The jump from conventional mobile phones to smartphones came accompanied by mobile OS. This is in charge to control all the programs and functions of the mobile. One of the most important OS and on which the application described in this document has been programmed, is Android.

The mobile market is dominated by this OS for many years, having only one strong competitor, iOS. Some of the strengths of Android versus iOS:

- Android is intended in open source code. This means that all of us can contribute to the development of the code. The knowledges about Android are shared in the net for all who want to develop an android app.

- Android is incorporated in almost all phone brands in the market, meanwhile iOS is only incorporated in apple devices.

- The prices of the devices of both OS are quite different. The cheapest iOS devices are around 500-600€ while the cheapest android devices are around 200€.

These reasons make Android the strongest mobile OS of the market, and the future forecast is that it continues.

On the other hand, android is gaining so much recognition due to some factors as the growth of the developer community, currently there are more than 3.800.000 applications in the Google Play Store. To develop an android app is an easy way to earn money so many people is trying their luck. Other reason is the digital transformation of companies. Nowadays having an application is synonymous with staying alive in the competitive mobile market. The customers want the possibility of doing the same as they would do in the establishments of the companies from home. The way to get this is through the applications.

Knowing all I have said, I would want to develop my own android app and I see the opportunity in this project. I could reach many people through android. By doing a research, I came to the conclusion of developing a health-related application, and in this way, I could learn a new programming language and at the same time help people. The purpose of the application is to research if it is possible to automatically measure the breathing frequency using only a mobile phone. In addition, I decided to develop a completely functional app for publish it in the Google Play Store.

Before start to program, I had to learn the Android basics through internet tutorials, and then I could start the development. The first step was a brainstorm, in order to imagine a first version of the application. I drew in a paper all the user interfaces of the views and wrote all the functionality that I wanted to include. Of course, during the development this first version was changing until get the current application.

The next step was starting to develop the application. Then I am going to resume all the views and their functionalities. In total it consists in sixteen views with different functionality each one of them.

Splash Screen:

This view is the first that appears always you enter the application. Is used to show the logo of the application, and maybe some information of the developers.

Log in screen:

This view is used to check the user credentials before show the app information. If is the first time you access to the app you will have to register as a new user.

Register Screen:

This view is used to register new users. They enter some personal data and create a new account. After register in the app you must activate your email before entering.

Activate Screen:

This view is used to activate the emails that have just registered. This step is necessary if you want to access the app with your account.

Recover Password Screen:

This view is used to reset your password when you forget it. Through your email you receive a recover code that you must entered with your new password.

Menu Screen:

This view is used to show all the sections of the app to the user when he enters the app. And it helps to navigate through the app. This screen is shown after the splash screen when there is an active session open, instead of Log in Screen.

Settings Screen:

This view is used to change the personal data of the user account. For example, the name, the email, the password, the profile image, and also to delete the account.

Calendar Screen:

This view is used to add a new event in the mobile calendar. If you want to remember that someday you should do exercise, you can use this functionality.

Contact Screen:

This view is used to send an email to the developers in order to ask a question, suggest some improvement to the app or report any issue that the user have found.

Help Screen:

This view is used to help the user with the app usage. It has a section about license information, a video tutorial about general usage of the app, and a video tutorial about how to take a good breathing frequency measure. Each one of these functionalities is a new view so this contain three more views.

Historic Screen:

This view is used to show all the measures that the user has taken and saved in the database.

Measure Screen:

This view is used to take a measure for fifteen seconds form the accelerometer sensor.

Result screen:

This view is used to show the result of the measure. I mean show in a graphic the breath signal and show how many breaths the user has taken for a minute.

After write all the views code, I have tested all the functionalities separately and together. When I am sure that all the app is working fine, I did the most difficult part of the app that is write the algorithm to take the measure and show the count.

I have done many tests as: use the accelerometer and the gravity sensor to take the measure, change the sensor delay until get the best function, change the labels scale of the graphic until get the best representation of the function, test the measure in different mobile devices, use a low-level filter to clean the noise of the function, calibrate the sensors, and so forth.

After all these tests, the best configuration to get the best function is: use only the accelerometer sensor, not with the gravity sensor, use the low-level filter to clean the noise of the function and set the delay sensor to SENSOR_DELAY_UI. In this way I have got an almost clear function. Now using the function, I had to say how many breaths the users takes, but this was not trivial. The problem was that the function already had some noise that makes impossible to difference between breaths and other movements.

In the measurement many factors influence: body palpitations, heartbeats, vibrations of the surface where you are laying on, notifications in the mobile which has vibration activated. All of these joined with the sensors that our devices have, is not enough to determine in an accurate way the breathing frequency. We would need to use more powerful sensors to get a good measure.

Índice de Contenido

Índice de Figuras

Índice de Tablas

1. INTRODUCCIÓN

Nos encontramos en una época en la que la tecnología avanza a un ritmo vertiginoso. A nivel social, actualmente podemos diferenciar dos ideas totalmente opuestas en cuánto a lo que la tecnología se refiere. Dentro de la generación más adulta, como pueden ser nuestros abuelos, no se podían imaginar que vivirían para ver cosas como los teléfonos móviles, que hoy en día son algo de los más cotidiano. Y por otro lado tenemos a las nuevas generaciones que no se pueden imaginar la vida sin estos avances tecnológicos.

Como se ha mencionado previamente uno de los avances tecnológicos con más impacto en la sociedad ha sido el teléfono móvil, surgiendo de la necesidad de comunicarnos entre las personas desde cualquier lugar y en cualquier momento. Éste, sería el objetivo inicial cuando aparecieron los móviles, sin embargo, hoy en día quizás la utilidad menos utilizada de los teléfonos móviles sea la de comunicarse a través de llamadas. Comunicarse sigue siendo el objetivo, pero de maneras muy diferentes.

Según la página de la asociación que engloba a la industria mundial del móvil (GSMA), en junio de 2018 hay 5.249.709.650 usuarios [1], de las 7.593.699.000 personas que hay en el mundo [2]. Esto significa que el 69,13% de la población tiene un teléfono móvil, y se prevé que este número siga creciendo.

Para conseguir alcanzar todas las funcionalidades que nos ofrecen los móviles, tuvieron que evolucionar muy rápidamente, dando lugar a los que conocemos como smartphone. Estos se consideran miniordenadores que nos posibilitan hacer casi cualquier cosa. Durante esta transición del teléfono móvil convencional al smartphone como no podía ser de otro modo, surgieron los sistemas operativos para gestionar dichos smartphones.

En concreto el 5 de noviembre de 2007 comenzó tal y como lo conocemos el sistema operativo móvil "Android", aunque hasta un año después, no aparecería incorporado en un teléfono móvil. [3]

Si no fue desde los inicios, poco tiempo necesitó Android para hacerse con el total control sobre la cuota de mercado móvil. Todavía hoy sigue manteniendo ese dominio, y no parece que la situación vaya a cambiar. Esto se debe a varios factores:

- Comparándolo con su gran competidor iOS, Android está enfocado a ser un sistema de código abierto. ¿Qué quiere decir esto? En resumidas cuentas, significa que facilita el acceso para que cualquier persona pueda desarrollar aplicaciones basadas en su sistema operativo.

- Facilita la interoperabilidad con otras plataformas.

- Android se encuentra presente en prácticamente todas las marcas de telefonía del mercado. Mientras que iOS está reducido a sus modelos.

- La diferencia de precios existente en las gamas de ambos SO es muy notable, estando la gama más baja de iOS entre 500€-600€, mientras que las gamas más bajas de Android se sitúan entre los 150€-200€.

- Pese a todos los anteriores motivos iOS sigue buscando el mercado en sus fieles seguidores, mientras que Android intenta abarcar la plenitud del mercado, y de esta manera no se prevé que haya cambios en cuanto al liderazgo del mercado por el momento.

A continuación, en la Fig.1. se muestra una gráfica estadística cuyos datos provienen del blog digital55 [4], en la que se aprecia el completo dominio de Android sobre la cuota de mercado en España en 2017.

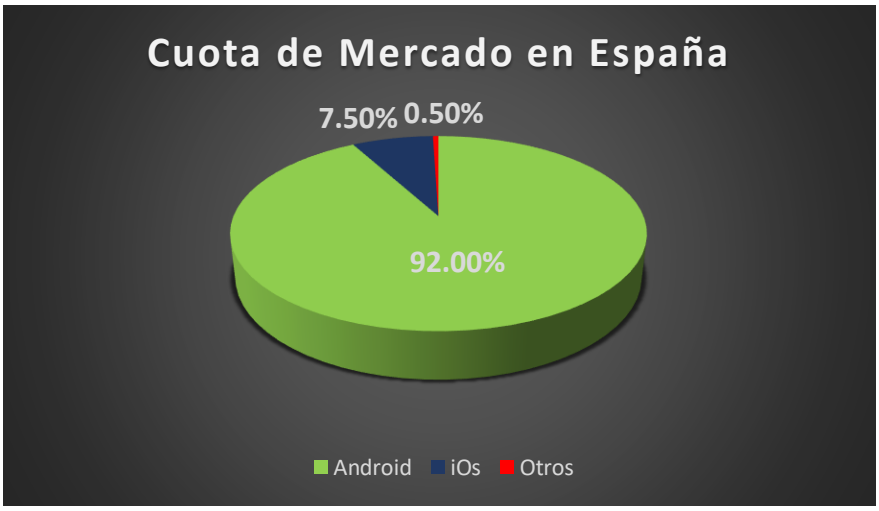


FIG. 1. CUOTA DE MERCADO MÓVIL EN ESPAÑA

Pero la situación descrita en la Fig.1. no es un caso aislado de nuestro país. En el resto de los países del mundo, aunque no tiene un dominio tan absoluto, todavía está bastante por encima de sus competidores. Los casos en los que iOS está más cercano a Android son los países con mayor poder adquisitivo. Haciendo referencia al motivo de la diferencia de precio en las gamas de cada SO que he mencionado con anterioridad.

Siguiendo el hilo de esta introducción, como se ha introducido previamente, la funcionalidad de los smartphones es muy diferente a sus inicios. Actualmente tienen infinidad de utilidades no solo la comunicativa. Y cada vez más y más se van integrando en la vida cotidiana hasta el punto de que ya los utilizamos para escuchar música, pagar nuestras compras, buscar cualquier información en internet, publicar en nuestras redes sociales, mantener contacto entre empresas y clientes, etc. Incluso tenemos aplicaciones que nos ayudan en aspectos para controlar el físico y la salud, como podría ser medir la frecuencia cardiaca a través de wereables.

Otro motivo por el que Android está ganando reconocimiento a nivel global, es por el crecimiento de la comunidad de desarrolladores, ya sea a nivel freelance o a nivel empresarial. En 2018, según el portal Statista [5], en el Google Play Store existen unas 3.800.000 aplicaciones. Este crecimiento se debe a que puede suponer una fuente de ingresos para cualquier persona, sin tener que invertir mucho a cambio. Lo principal que requiere el desarrollo de una aplicación a modo de inversión es tiempo. Si tienes tiempo

y buenas ideas eres la persona indicada para desarrollar una aplicación ya sea por motivos altruistas o por motivos económicos. Más adelante se enumerarán algunas formas de sacar beneficios a una aplicación.

Pero si se tuviera que decir cuál es el principal motivo por el que Android está teniendo tanta acogida y tanto crecimiento, sin duda sería por la transformación digital de las empresas. Los clientes de las empresas demandan poder hacer todo tipo de gestiones sin tener que moverse de casa y en el menor tiempo posible. Esto se consigue con las aplicaciones móviles, tanta importancia supone esto, que ahora mismo para una empresa disponer de una aplicación móvil para sus clientes es sinónimo de sobrevivir en el competitivo mercado que existe. La aplicación puede ser para ofrecer los mismos servicios que ofrecerían desde sus locales u oficinas, o simplemente para publicidad sobre su empresa, el motivo no importa realmente, lo importante es que el cliente tenga a mano la información de tu empresa.

Por todos estos motivos, el sistema elegido para el desarrollo de la aplicación descrita en este documento es Android, pudiendo llegar de esta manera a una mayor cantidad de personas como hemos podido ver en la Fig.1. Además, la facilidad que supone que en internet se pueda encontrar una gran cantidad de documentación, tutoriales y ejemplos sobre el desarrollo Android también facilita que se despierte interés por este tema. Incluso disponemos de la gran comunidad Android Developers que recoge la explicación de toda la funcionalidad y posibilidades que le puedes incluir a tu aplicación.

2. MOTIVACIONES Y OBJETIVOS

En cuanto a mis motivaciones para realizar este proyecto, el principal motivo es que, a lo largo del desarrollo de mi carrera universitaria con las asignaturas de programación y desarrollo web, se despertó mi interés por aprender programación Android. Teniendo este objetivo en mente no había mejor manera de llevarlo a cabo que aprovechar este proyecto.

Puesto que el desarrollo Android está basado en el lenguaje Java para la funcionalidad y la lógica de la aplicación, y en el lenguaje XML para el diseño de la interfaz de usuario, era un aliciente más para llevarlo a cabo, ya que he podido adquirir conocimientos sobre ambos lenguajes gracias a algunas asignaturas de la carrera y de esta manera me resultaría más fácil que si partiera sin ninguna base. Aunque he de decir que, pese a esto, antes de comenzar el proyecto no tenía ningún tipo de conocimiento sobre Android.

Tras un análisis tratando de encontrar aplicaciones parecidas a la descrita en esta memoria, he podido ver que existe alguna como *"Breath Counter 3.0"*, que lejos de medir de manera automática las respiraciones, consiste en pulsar un botón con cada respiración. Otras como *"Breath Sounds (Lungs Sounds)"*, tratan de medirlas mediante el sonido, y la más parecida que he encontrado sería *"Breath Rate"*. Ésta última no he podido probarla ya que cuesta dinero descargarla, pero por algún comentario que tiene, entiendo que trata de medir a partir de unas configuraciones por parte del usuario.

Con el análisis anterior en mente, y tratando de hacer algo diferente, el objetivo final que me propuse ha sido: *"Analizar si es posible o no es posible medir de manera precisa la frecuencia respiratoria utilizando el acelerómetro del móvil, y sin la ayuda de ningún wereable*". En complemento a esto, me propuse también el añadido de completar la aplicación con la mayoría de las funcionalidades posibles que nos ofrece Android siempre y cuando tuvieran relación con el objetivo global, y presentarla como una aplicación totalmente funcional para publicarla en la Google Play Store.

3. PLANIFICACIÓN Y PRESUPUESTO

3.1. PLANIFICACIÓN Y DESCRIPCIÓN DE LAS TAREAS

En este apartado se presenta como se ha desarrollado el proyecto en el tiempo, las distintas tareas que han sido requeridas para llevarlo a cabo, y la descripción de cada una de ellas.

La Fig.2. muestra el diagrama de Gantt con las diferentes tareas llevadas a cabo y la duración que ha supuesto cada una de ellas.

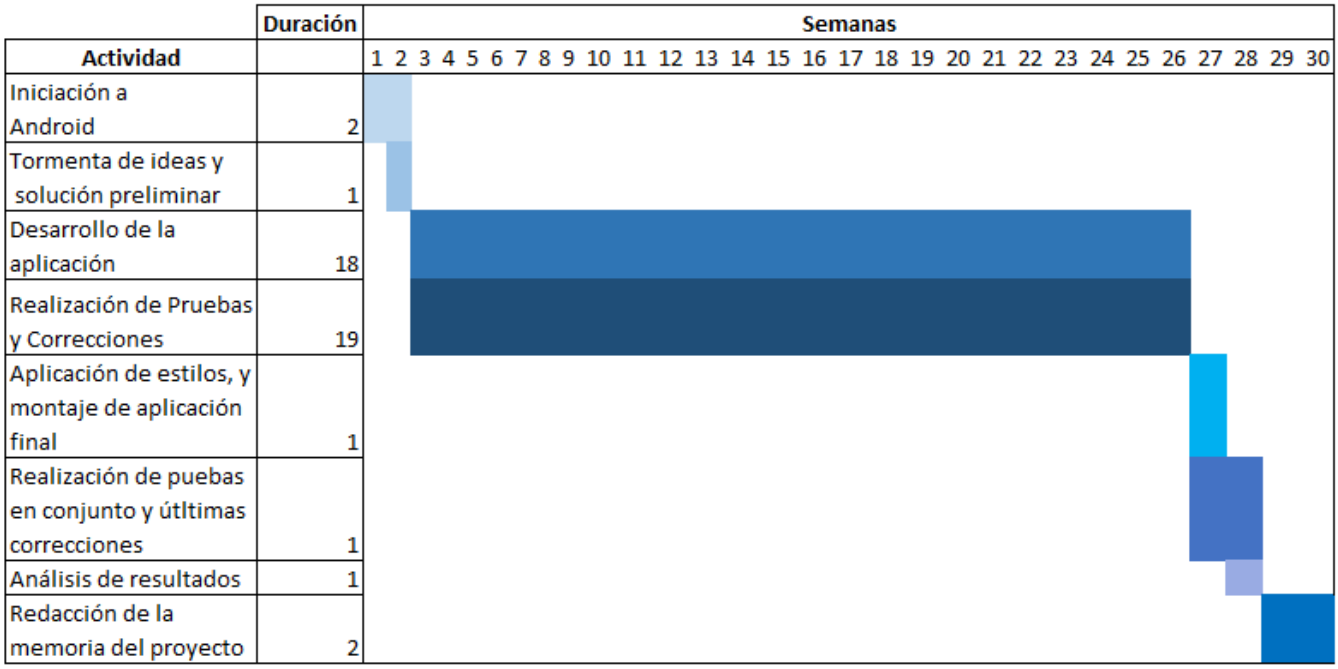


FIG. 2. DIAGRAMA DE GANTT DEL PROYECTO

Como se puede apreciar en la Fig.2. el proyecto ha tenido una duración de 30 semanas, o lo que es equivalente, 7 meses. Además, consta de 8 tareas. En este caso la duración de lo que es el desarrollo de la aplicación coincide con las pruebas y correcciones. Esto se debe a como se ha ido desarrollando la aplicación, es decir, para cada una de las vistas de la aplicación primero se creaba y después se hacían las pruebas de manera aislada a la aplicación principal, y sin aplicar estilos ni recursos finales.

Actividades

- **Iniciación a Android**: la primera tarea llevada a cabo para comenzar el proyecto fue preparar el entorno de trabajo, es decir, instalar los programas necesarios para desarrollar [6]. A continuación, puesto que se partía sin tener conocimientos sobre Android, se tuvo que llevar a cabo una segunda tarea de lo que sería aprender los aspectos básicos de Android, a través de tutoriales de iniciación como los siguientes:
    a. Tutorial Android desde cero [7].
    b. Curso Android Studio [8].
    c. Canal de Youtube sobre Android Asesorías Maldonado [9].

  Desarrollando una miniaplicación para probar los conocimientos adquiridos con los tutoriales anteriores. (Duración 2 semanas).

- **Tormenta de ideas y solución preliminar**: esta tarea consistió en plasmar en un papel las primeras ideas sobre como se pensaba que iba a ser la aplicación, el diseño de las diferentes pantallas que habría en la aplicación, y las funcionalidades que incorporarían. Posteriormente durante el desarrollo, esta primera idea se ha ido modificando, quitando y añadiendo cosas. (Duración 1 semana).

- **Desarrollo de la aplicación**: esta tarea junto a la siguiente se llevan el grueso del tiempo del proyecto, ya que consta en programar la interfaz de usuario de cada pantalla y programar la lógica interna de cada pantalla. (Duración 23 semanas).

- **Realización de pruebas y correcciones**: esta tarea junto a la anterior se lleva el grueso del tiempo del proyecto, ya que consta en probar de manera individual cada una de las pantallas que compondrán la aplicación, y corrección de los errores encontrados. (Duración 23 semanas).

- **Aplicación de estilos, y montaje de aplicación final**: esta tarea consiste en aplicar los estilos a la aplicación. Esto es, tipo de letra de los textos, diseño de imágenes utilizadas, etc. Y finalmente juntar cada una de las pantallas que se han creado por separado en una única aplicación global. (Duración 1 semana).

- **Realización de pruebas en conjunto y últimas correcciones**: esta tarea consiste en probar la funcionalidad de la aplicación de manera conjunta tras unir todas las pantallas en una sola. Y solucionar cualquier problema que pueda surgir en el correcto funcionamiento global de la aplicación. (Duración 2 semana).

- **Análisis de resultados**: esta tarea consiste en probar la funcionalidad de medir la frecuencia respiratoria, hacer modificaciones pertinentes para conseguir una medida válida, y finalmente crear una conclusión en base a los datos obtenidos. (Duración 1 semana).

- **Redacción de la memoria del proyecto**: esta tarea consiste en redactar este documento, recogiendo toda la información sobre el desarrollo del proyecto. La duración de ésta debería ser equivalente a la totalidad del tiempo del proyecto ya que desde el inicio se ha ido guardando la información necesaria para cumplimentarlo. Pero, sin embargo, refleja una duración de dos semanas que sería lo equivalente a la redacción real de la memoria. (Duración 2 semanas).

3.2. PRESUPUESTO Y MATERIALES UTILIZADOS

En este apartado se detallan todos los materiales que han sido necesarios para la cumplimentación del proyecto, y además el presupuesto que éste ha supuesto.

Recursos Humanos

Elemento	Dedicación (horas)	Coste/Hora	Gasto Total
Sergio García Águila	700	14,06€	9.842€
Mario Muñoz Organero	10	18,23€	182,23€
5 Ayudantes para pruebas	10	0€	0€

TABLA 1. RECURSOS HUMANOS

Los datos de la tabla se corresponden a las siguientes suposiciones:

a. Sergio García Águila: ingeniero junior, encargado de desarrollar la aplicación en su totalidad, coste por hora calculado por 27.000€ de media anuales.
   Dedicación total → ((4h al día * 5 días a la semana) + 5 h fines de semana) * 30 semanas = 750 horas.
   A las 750 le restamos 2 semanas de vacaciones → 750 - 50 = 700 horas.

b. Mario Muñoz Organero: ingeniero senior, encargado de supervisar el avance del proyecto, coste por hora calculado por 35.000€ de media anuales. Tareas desempeñadas: tutorías y revisión memoria del proyecto y aplicación.

c. Ayudantes para pruebas: voluntarios encargados de concluir si se trata de una aplicación intuitiva o no, mediante interacciones con la interfaz de usuario, y además probar la funcionalidad de tomar la frecuencia respiratoria.

Recursos Materiales

Elemento	Coste
Android Studio	0
Adobe Photoshop CS6	0
WonderShare Filmora	0
Microsoft Word 365	0
Microsoft Excel 365	0
DropBox	0

TABLA 2. RECURSOS SOFTWARE

- **Android Studio:** es el entorno/programa utilizado para escribir el código de la aplicación.

- **Adobe Photoshop CS6:** programa utilizado para la creación de las imágenes contenidas en la aplicación, y la edición de estas.

- **WonderShare Filmora:** programa utilizado para la edición de los videotutoriales incluidos en la aplicación.

- **Microsoft Word 365:** programa utilizado para la redacción de la memoria del proyecto.

- **Microsoft Excel 365:** programa utilizado para el diseño del diagrama de Gantt contenido en esta memoria.

- **Dropbox:** programa utilizado para almacenar una copia de seguridad del proyecto.

Elemento	Coste
Ordenador Portátil	750€
Smartphone x3	279€

TABLA 3. RECURSOS HARDWARE

- **Ordenador Portátil:** ordenador personal que contiene todos los recursos de software indicados anteriormente.

- **Smartphone x3:** se ha utilizado principalmente mi smartphone personal para realizar las pruebas de código y todo lo relacionado con la aplicación. Los otros dos smartphones indicados son los utilizados para probar la aplicación en distintos tamaños de pantalla y distintos modelos de fabricantes.

Elemento	Coste
Recursos Humanos	10.024,23€
Recursos Materiales	
Software	0€
Hardware	1.029€
Total	**11.053,23€**

TABLA 4. COSTES TOTALES

4. ESTADO DEL ARTE

En esta sección se van a introducir los aspectos básicos necesarios para empezar a programar una aplicación Android y aspectos que hay que tener en cuenta para una buena programación.

4.1. ESTRUCTURA DE UNA APLICACIÓN

Lo primero será definir la estructura de la aplicación y sus diferentes directorios. La Fig.3. muestra en concreto la estructura de la aplicación que se ha desarrollado. Una estructura puede contener más de los directorios mostrados aquí, pero para centrar esta memoria en lo que es la aplicación sólo se describirán los aquí mostrados.

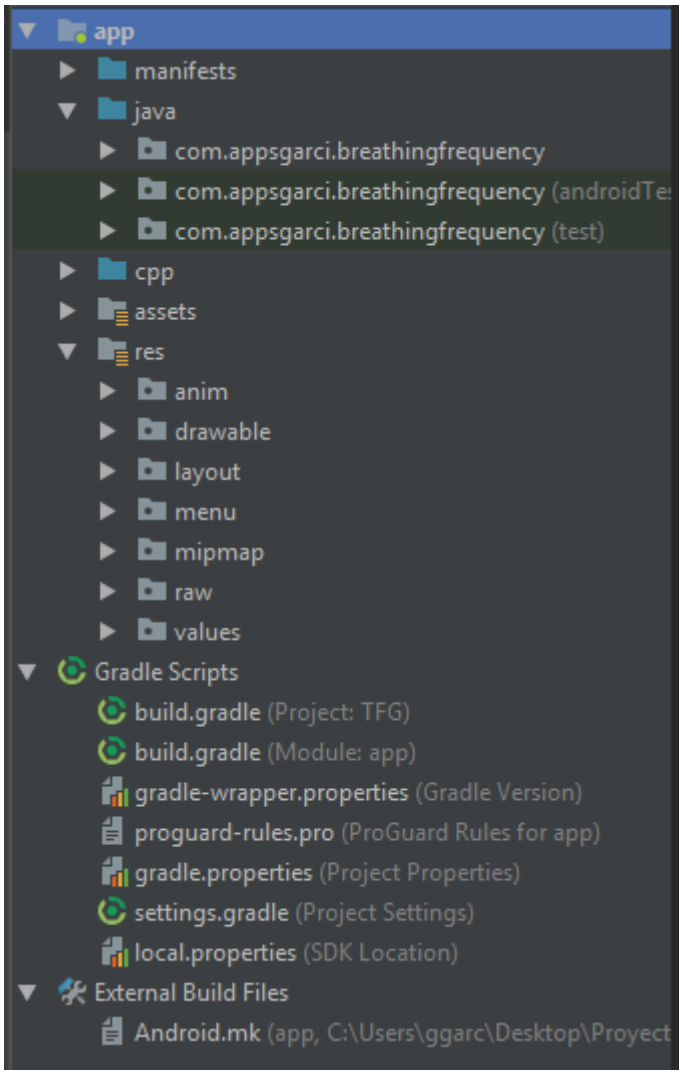


FIG. 3. ARQUITECTURA DE UNA APLICACIÓN

- **manifests:** contiene el archivo manifest de la aplicación. En él se recoge el paquete de la aplicación, los permisos que necesita la aplicación para funcionar correctamente**,** y todas las activities que contiene la aplicación y sus configuraciones.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.appsgarci.breathingfrequency">
```

FIG. 4. PAQUETE DE APLICACIÓN

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
```

FIG. 5. PAQUETE DE APLICACIÓN

```
<activity
    android:name=".SplashScreenActivity"
    android:screenOrientation="sensorPortrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

FIG. 6. ACTIVITY PRINCIPAL Y ORIENTACION

En la Fig.6. podemos ver la cual es la activity que se muestra nada más abrir la aplicación como indica las líneas:

<action android:name=”android.intent.action.MAIN”/>

<category android:name=”android.intent.category.LAUNCHER”/>

Y además estará siempre en orientación vertical, como indica el parámetro:

android.screenOrientation=”sensorPortrait”

Este parámetro es el que se ha utilizado para que toda la aplicación funcione en orientación vertical. Si se quisiera cambiar a orientación horizontal, se cambiaría el valor del parámetro por sensorLandscape.

- **java:** contiene los archivos .java que poseen la lógica de cada una de las activities, y también las clases auxiliares utilizadas para funcionalidad extra.

- **cpp:** también conocida con “jni” contiene archivos en lenguaje C o C++, utilizados por ejemplo para funciones nativas. En este caso se han utilizado dos archivos [10]:

- keys.c: contiene cuatro funciones, una que devuelve la contraseña de la base de datos, otra dos que devuelven dos direcciones de correo electrónico, y otra que devuelve la contraseña de una de las direcciones de correo anteriores.
- Android.mk [11]: este archivo es el encargado de recopilar la información contenida en el archivo keys.c y montarla como una librería nativa del proyecto.

Estos archivos han sido necesarios para cifrar información delicada. Si pusiéramos dicha información directamente en los archivos Java, cualquiera podría sacar el código de la aplicación a partir del .apk y obtenerla. Sin embargo, de esta manera al ir en código C no se puede desencriptar. De igual manera para las direcciones de correo y la contraseña que se necesitaban que fueran en código.

- **assets:** contiene archivos de fuentes personalizadas para aplicar a los textos de la aplicación.
- **res:** contiene todos los archivos relacionados con los recursos de la aplicación. Imágenes, iconos, vistas de las activities, animaciones, etc.
  - **anim:** contiene los archivos de animaciones que luego puedes aplicar a los elementos del activity. En este caso se han utilizado para definir un efecto de transición al cambiar de activity. Se utilizan con la siguiente línea de código:

```
overridePendingTransition(R.anim.fade_in,R.anim.fade_out);
```

  - **drawable:** contiene archivos .xml con configuraciones de estilos, contiene las imágenes de la aplicación y contiene los iconos de la aplicación.

  - **layout:** contiene los archivos .xml que definen la interfaz gráfica de cada una de las activities.

- **menu:** contiene archivos .xml que definen elementos de tipo menú. En este caso se hs utilizado para definir el menú desplegable de la barra de navegación.

- **mipmap:** contiene los iconos de la aplicación, es decir, los que se verían en el escritorio de tu smartphone cuando instalas la aplicación.

- **raw:** contiene el resto de los archivos que no encajarían en las demás carpetas. En este caso contiene un archivo de sonido para indicar que la aplicación ha terminado la medida de la frecuencia respiratoria, y dos archivos de vídeo con los videotutoriales de la aplicación.

- **values:** contiene tres archivos: colors.xml, strings.xml, styles.xml. El primero de ellos contiene los valores hexadecimales de los colores que se utilizan en la aplicación. El segundo de ellos contiene todos los textos de la aplicación, y el último de ellos contiene todos los estilos de la aplicación.

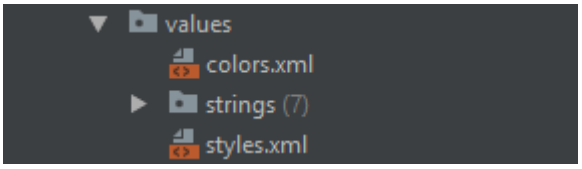


FIG. 7. CARPETA VALUES

Este directorio podríamos decir que es el de buenas prácticas. ¿Por qué?, esto se debe a que estos tres archivos son opcionales, no hay por qué utilizarlos. Se podrían poner directamente los colores, textos y estilos en el código, pero ante cualquier cambio, tendríamos que cambiar todos y cada uno de los sitios donde los hubiéramos escrito. De esta forma solo cambiamos estos tres archivos y se cambiaría en toda la aplicación. Además, en cuanto al archivo strings.xml si contiene todos los textos, posibilita la opción de traducir la aplicación.

- Por último tenemos el archivo build.gradle (Module app) dentro del apartado Gradle Scripts. Este archivo contiene información importante de la aplicación. Por ejemplo, contiene la versión del sdk actual de la aplicación, contiene el id de la aplicación, la versión mínima del sdk que acepta (ésta será la mínima versión Android que soportará la aplicación), y define todas las librerías externas que utiliza la aplicación.

4.2. CICLO DE VIDA DE UN ACTIVITY

Una vez tenemos clara la estructura de una aplicación Android, toca definir el ciclo de vida de un activity. Esto es muy importante ya que hay que tenerlo en cuenta a la hora de definir las llamadas a las funciones, para definir en qué momento realizar cada acción.

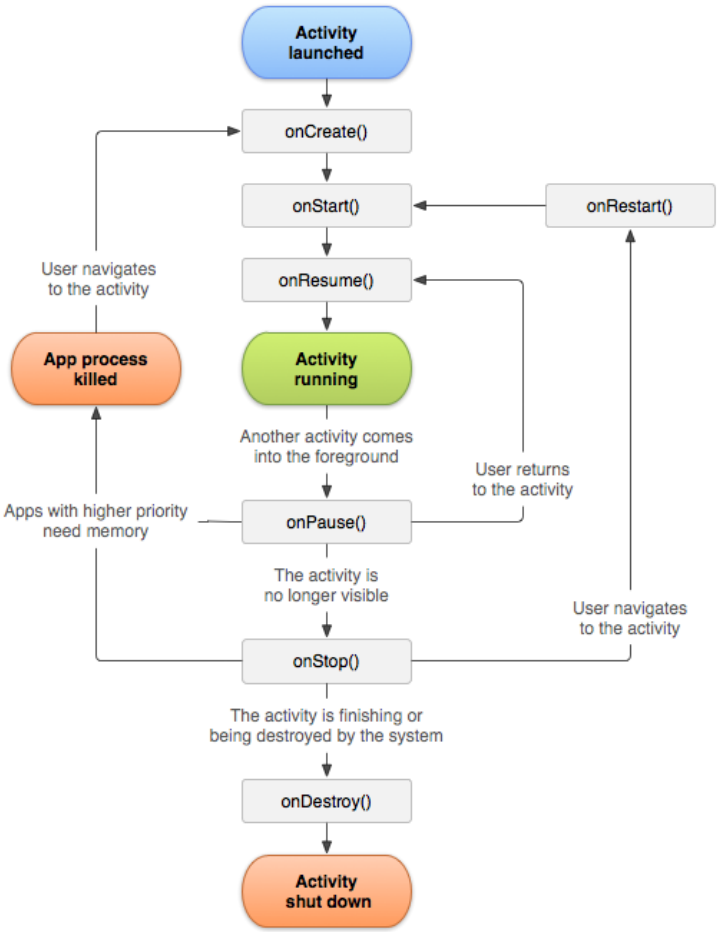


FIG. 8. CICLO DE VIDA DE UN ACTIVITY ***Fuente: Android Developers [12]***

Como podemos ver en la Fig.8. cuando un activity es llamado se entra al método onCreate(), éste es el estado principal y donde definiremos las variables y llamaremos a las funciones que sean necesarias. También es el punto donde se asocia un activity con su layout.

Otros de los estados importantes serían el método onPause() y onStop(), éstos se diferencian entre sí porque el método onPause() se llama inmediatamente después de que un activity diferente al actual pase a primer plano, y el método onStop() se llama cuando estando en onPause() transcurre un tiempo prudencial para dar por acabado el activity. Tras un tiempo después de estar en onStop(), si siguiera sin haber actividad se llamaría al método onDestroy() y se destruiría finalmente el activity. Este último método suele ser llamado por el gestor de memoria del smartphone.

Este flujo de estados es importante si queremos hacer un buen uso de la memoria del teléfono, pero hay que tener cuidado de donde se hacen las operaciones ya que podríamos estar acabando el activity sin darnos cuenta, antes de hacer todas las operaciones.

4.3. BUENAS PRÁCTICAS

En este apartado voy a describir algunas de las buenas prácticas que es aconsejable llevar a cabo durante el desarrollo de una aplicación. Todas las que se van a describir las he tenido que ir aplicando para facilitarme la tarea, ya que previa aplicación, me costaba mucho reutilizar código, entender flujo de ejecución de código, etc.

- En primer lugar, y este no aplica solo para Android, estaría el comentar de manera detallada el código que vamos escribiendo. Esto es muy importante para poder reutilizarlo, y entender por qué en el momento que estamos escribiendo esa parte lo hicimos de ese modo. Puede pasar que tengas que parar el desarrollo por unas vacaciones, o cualquier otro motivo, y si cuando lo retomes quieres entender lo que llevabas hecho será mejor que en su día lo comentaras.

- Otra buena práctica como se ha comentado en el apartado 4.1, es hacer uso de los archivos colors.xml, styles.xml y strings.xml. Si se definen todos los colores dentro del primer archivo no solo no tendrás que cambiar en varios puntos de código cada vez que se requiera una modificación, sino que también se evitará poner diferentes tonalidades. Hay que ser coherente con los colores que se utilizan, y si se quiere reutilizar el mismo, ésta es la mejor manera.

  En cuanto al archivo styles.xml contiene un elemento muy importante que es el tema de la aplicación, por defecto viene el tema DarkActionBar, el cual crea la barra de navegación superior donde normalmente aparece el menú de opciones. Si se quiere quitar esta barra de navegación tenemos que aplicar el estilo NoActionBar.

```
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="android:windowFullscreen">true</item>
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:statusBarColor">@android:color/transparent</item>
</style>
```

FIG. 9. TEMA DE LA APLICACIÓN

En la Fig.9. Podemos observar cómo se ha quitado la barra de navegación, y además aparece un parámetro que se llama android.windowFullscreen a valor true. Este nos indica que cuando se muestre la aplicación la barra de notificaciones se ocultará, y solo se mostrará si arrastramos hacia abajo en lo alto de la pantalla.

El último de los tres archivos sería strings.xml, si recogemos aquí todos los textos de la aplicación nos resultará más fácil hacer cualquier cambio, además de no tener que estar escribiendo constantemente lo mismo. En este caso además ha servido para traducir la aplicación a otros idiomas. Esto se consigue creando varias carpetas a la altura de values, con el código de idioma, tal y como muestra la Fig.10.

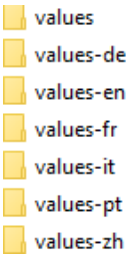


FIG. 10. IDIOMAS DE APLICACIÓN

En este caso se ha traducido la aplicación a siete idiomas diferentes: por defecto que es el español, inglés, alemán, francés, italiano, portugués y chino simplificado. Dentro de cada una de las carpetas de la Fig.10 hay un archivo strings.xml con los textos en cada idioma.

- Siguiendo con las buenas prácticas, otro consejo es separar del código principal las funciones. Para que el código sea más legible, conviene que en el método onCreate() hagamos una llamada a una función que declaramos fuera de éste. Si lo hacemos de esta manera el código estará más limpio que si hacemos toda la lógica dentro de un solo método. Incluso de esta forma estamos favoreciendo la reutilización de código, ya que si en otro activity necesitamos hacer algo de la misma manera podemos llamar a esta misma función sin volver a escribirla entera.

- Utilizar una programación multihilo también nos ayuda a hacer un código más eficiente en cuanto a gestión de memoria se refiere. Si hacemos todas las funciones en el hilo principal y alguna tarea es más pesada de la cuenta, podría bloquear la interfaz de usuario, algo que no es nada deseable. Para enfrentarnos a este problema disponemos de los hilos o Threads, y de AsyncTask, que nos permite definir tareas en hilos diferentes y ejecutarlos de manera asíncrona.

- Otro aspecto importante, es eliminar recursos que no utilicemos, puede ser que en un principio utilizáramos una imagen que ahora hemos decidido cambiar, o un texto que ya no queremos usar, en ese caso lo mejor es eliminarlo. Para esta tarea Android Studio nos da una opción que analiza el código en busca de

recursos sin usar. Simplemente tendremos que hacer click derecho en alguno de nuestros archivos abiertos > Refractor > Remove Unused Resources > Preview. Escogemos preview para ver cuántos recursos están sin utilizar y de esta forma podemos elegir cual no nos hace falta y cuales tenemos pensado usar en el futuro. Otra forma de optimizar recursos es que a la misma altura que la carpeta drawable, creemos otras carpetas de la siguiente manera:

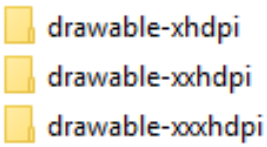


FIG. 11. DENSIDAD DE IMÁGENES

Las carpetas de la Fig.11. hacen referencia a la densidad por pixel de las imágenes. Así pues, si tenemos alguna imagen que ocupa bastante espacio, lo mejor es meterla en xxxhdpi, de esta manera la aplicación hará una mejor gestión de memoria cuando tenga que utilizarla. Si además la convertimos a la extensión .webp también ayudará a la memoria de la aplicación. Esto último se puede hacer dentro de Android Studio, botón derecho sobre la imagen en cuestión > convertir a webp.

- Por último y para cerrar este apartado, el último consejo es que, a la hora de diseñar la interfaz gráfica de cada activity, se usen medidas relativas. Esto se refiere a que cuando se defina el tamaño de una imagen, o se coloquen los diferentes elementos sobre el layout, se referencien entre sí, o con los marcos del teléfono. Si al colocar un texto se quiere en el centro de la pantalla y a una distancia del borde, puesto que no sabes en que móvil se estará ejecutando, no trates de imaginar cuanto hay que poner para colocarlo, lo mejor es hacer referencia al marco del teléfono y al centro de la pantalla. Para esto existe un tipo de layout denominado RelativeLayout que nos permite un control total sobre los elementos que colocamos en él.

4.4. ELEMENTOS BÁSICOS DE LA APLICACIÓN

En este apartado se va a describir todos los elementos que se han ido utilizando a lo largo del proyecto. De esta manera cuando lleguemos a la descripción de cada activity será más fácil la comprensión de la utilidad de cada elemento.

Elementos

- **TextView:** se trata de un cuadro de texto donde pondremos cualquier información o literal que queremos que aparezca en el activity. Son estáticos y el usuario no interactúa con ellos.

- **EditText:** se trata de un elemento pensado para que el usuario introduzca un texto que utilizaremos después para la lógica. El uso más conocido de estos es en los formularios. Donde se rellenan unos campos, ya sean de datos de registro, o datos de logado y posteriormente se envían.

- **Button:** se trata de los botones que se utilizan en la interfaz para interactuar con el usuario.

- **Spinner:** se trata de las listas desplegables para elegir una opción, y en función de ésta, escoger un camino u otro de la lógica.

- **ProgressBar:** se trata de una barra de progreso que se utiliza para hacer ver al usuario que se están haciendo operaciones de en segundo plano.

- **SeekBar:** se trata de una barra como las utilizadas en los reproductores de música y vídeo. Ésta va ligada a la duración de un elemento mediaplayer asociado a una canción o un vídeo. Se puede hacer operaciones sobre ella como avanzar el progreso, pausarlo, reanudarlo, etc.

- **Thumbnail:** se trata de una miniatura de una imagen, por ejemplo, para mostrar la imagen de perfil.

- **ListView:** se trata de un elemento que nos permite agregarle otros elementos y disponerlos a modo de lista ordenada.

- **ScrollView:** se trata de un elemento que nos permite hacer scrollables a los elementos que contenga. Esto quiere decir, que, si no entraran todos los elementos dentro de la pantalla del dispositivo, si deslizamos en la pantalla iran apareciendo los que no estuvieran a la vista.

- **ImageView:** se trata de un elemento que permite mostrar imágenes en la pantalla del dispositivo.

- **Toolbar:** se trata de un elemento que se utiliza para sobrescribir la barra de navegación y darle otro aspecto y funcionalidad diferentes.

- **AppBarLayout:** normalmente el contenedor de una toolbar, y se utiliza para darle funcionalidad a ésta.

- **DatePicker:** se trata de un elemento que permite mostrar un calendario para que el usuario introduzca una fecha.

- **TimePicker:** se trata de un elemento que permite mostrar un reloj para que el usuario introduzca una hora.

- **RadioButton:** se trata de un elemento que nos da la posibilidad de elegir entre opciones ya predeterminadas, haciendo click en ellas.

- **Toast:** se trata de un elemento que nos permite mostrar una información en un bocadillo emergente. El usuario no interactúa con él.

- **Dialog:** se trata de una ventana emergente que se muestra al usuario para que interactúe con ella.

- **GraphView:** se trata de un elemento que nos permite pintar un gráfico a partir de unos valores.

4.5. ¿CÓMO RENTABILIZAR UNA APLICACIÓN?

En este apartado se van a introducir algunas de las maneras de ganar dinero mediante el desarrollo de aplicaciones.

- La forma más fácil, y quizás menos atractiva de cara al usuario, es establecer un precio de descarga en la App Store. De esta manera estarás ganando dinero con cada descarga que se realice.

- Otra forma, que es utilizada sobre todo en juegos para smartphone, es poner compras in-game. Esto se trata de ofrecer mejoras en el juego o la aplicación a cambio de dinero. Esta manera es muy parecida a establecer una versión premium de la aplicación. De esta manera ofrecemos más contenido de la aplicación en la versión de pago o premium.

- La manera más conocida quizás es la de introducir publicidad en tu aplicación. Esta forma no es muy atractiva de cara al usuario, pero si se elige bien donde establecer la publicidad y el tipo de publicidad que se muestra puede ser una buena manera de rentabilizar tu aplicación.

- La manera más rentable a mi parecer, aunque también la más complicada de llevar a cabo es conseguir una asociación con empresas o comercios, que te ofrezcan una cantidad de dinero a cambio de estar presentes en tu aplicación. Esta forma va ligada en cierto modo a la última manera que expongo en este apartado. Ya que si consigues un alto volumen de usuarios es más fácil que las empresas quieran aparecer en tu aplicación.

- La última manera de ganar dinero con tu aplicación sería intentar conseguir un altísimo volumen de usuarios a cambio de establecer todo el contenido de manera gratuita, y de este modo llamar la atención de empresas que se asocien contigo, o conseguir vender la aplicación por un precio [13].

4.6. LEGISLACIÓN Y LICENCIAS

En este apartado se va a hablar sobre aspectos legales que hay que tener en cuanta si queremos desarrollar una aplicación Android y el tipo de licencias que rigen recursos que se han utilizado para la aplicación.

Cuando desarrollamos una aplicación móvil tenemos que tener presentes algunos puntos legales que debemos cumplir para evitar posibles sanciones en el futuro. Algunos de estos aspectos para tener en cuenta se listan a continuación:

- Protección de contenido según la edad. Hay que informar debidamente al usuario de una aplicación de la edad mínima requerida para poder usar la aplicación según el contenido que tenga. Google Play Store cuenta con la clasificación por edades PEGI.

- Hay que solicitar los permisos necesarios al usuario para que esté completamente informado sobre que funcionalidades de su dispositivo son utilizadas. Esto es casi inherente al desarrollo Android ya que, si los permisos no son aceptados por el usuario, la funcionalidad que hiciera uso de ellos no funciona.

- Otro punto para tener en cuenta es informar sobre las condiciones de uso de la aplicación. En caso de que los datos del usuario vayan a ser tratados o utilizados de algún modo, debe informársele y hacer que acepte un formulario. La nueva ley de GDPR sobre la protección de datos personales se encarga de ello.

- Por último, hay tener cuidado con los recursos que se utilizan y las licencias que posean. Es frecuente que cuando creas un contenido, ya sea imágenes, fuentes, estilos, aplicaciones y otros, se protejan ante el plagio mediante licencias. Este tipo de recursos no se pueden utilizar salvo que tengas permiso del creador. Para este cometido existen licencias de código abierto que nos permiten integrar recursos en nuestros proyectos sin peligro de sanción. Algunas de estas licencias que se han utilizado en la aplicación son: apache 2.0, MIT y OFL.

5. DESCRIPCIÓN DE LA APLICACIÓN

En esta sección se describirá todo lo relacionado con las consideraciones que se han tenido en cuenta para su desarrollo, el aspecto de cada una de las activities, y la funcionalidad de estas.

5.1. CONSIDERACIONES DE DISEÑO

Primeramente, antes de comenzar a describir cada una de las pantallas de la aplicación, y sus funcionalidades, conviene exponer algunas consideraciones que se han tenido en cuenta para el diseño, y de esta manera se entenderá todo mejor.

- La primera consideración es el diseño de la BD. En este caso se ha decidido optar por SQLite en vez de una BD externa. SQLite funciona como un archivo interno en el dispositivo. Para la funcionalidad de esta aplicación es más que suficiente de esta manera. Para conocer el uso de SQLite se ha consultado el tutorial del blog de sgoliver [14], el cual está dividido en tres partes. Para añadir seguridad a la BD se ha utilizado la librería SQLCipher [15], y se ha consultado el tutorial en Youtube del usuario EDMT Dev [16].

  La estructura de la BD consta de tres tablas: una para almacenar usuarios (nombre, email, contraseña, imagen de perfil, flag email activo, código de recuperación de contraseña y token de sesión), otra tabla de imágenes de medidas (usuario, imagen, actividad, fecha), en la que se guarda el histórico de cada usuario. Y por último una tabla que contiene todas las preguntas frecuentes para mostrarlas en la pantalla FaqsScreen (titulo, pregunta, respuesta).

- Otra consideración sobre la aplicación es que se permite registrar emails inventados, siempre y cuando tengan el formato whatever@whatever.com. Eso sí, en el caso de que se utilice un email inventado se dejará de tener

acceso a la funcionalidad de recuperar contraseña, pero el resto de las funcionalidades irán del mismo modo.

- Como se ha adelantado en otra sección, la aplicación tiene que funcionar con la pantalla del dispositivo en sentido vertical para todas las activities.

- Para la detección del idioma seleccionado y si hay sesión activa, se ha decidido usar las sharedpreferences [17]. Las sharedpreferences, es un archivo interno de la aplicación donde se guardan configuraciones. Podemos hacer uso de ellas para almacenar la configuración del usuario de cara a la aplicación. En este caso la primera vez que se accede a la aplicación coge el idioma predeterminado del dispositivo, si el usuario cambia dentro de la aplicación el idioma, éste se guarda en la sharedpreferences y permanecerá ahí hasta un nuevo cambio, o la eliminación de la aplicación. Por el lado del token de sesión, se almacena en el momento que se hace un inicio de sesión exitoso, y sólo se borra en caso de cerrar sesión de manera manual en la aplicación.

- En cuanto a la cantidad de imágenes que se puede guardar en BD para cada cliente, se ha decidido que sólo se puedan almacenar cinco. Ya que es más que suficiente para la funcionalidad de la aplicación.

- La versión mínima soportada por la aplicación es la sdk 23, que corresponde a Android 6.0 Marshmallow, aunque está pensada para a partir de la sdk 24, que corresponde a Android 7.0 Nougat. El motivo de esta decisión es que ya no quedan muchos dispositivos por debajo de esta versión, y además se utilizan funciones para las nuevas versiones.

- Para todos los textos de la aplicación se ha utilizado la fuente open source Expletus Sans [18].

- Para la aplicación de estilos a la ProgressBar de la SplashScreen, para todos los EditText presentes en la aplicación y para los Thumbnails de las imágenes de perfil, se ha utilizado el API de Bootstrap para Android [19].

- Los iconos utilizados en la aplicación, tanto los del Spinner personalizado, como los de las Toolbars, se han obtenido del propio Android Studio que provee de un banco de iconos open source, bajo la licencia apache 2.0.

- Todos los recursos, imágenes, sonido y vídeos, que contiene la aplicación, han sido diseñadas por mi utilizando Photoshop CS6. A excepción de los iconos como se ha comentado en punto anterior, y el logo tiene como base una imagen sacada del repositorio openSource Pixabay [20].

- Se ha decidido traducir la aplicación a siete idiomas porque está pensada para poder ser usada a nivel internacional, y en el mayor número de países posibles.

- Como se vió en la Fig.5. de la sección de estructura del proyecto, se ha utilizado los permisos de internet y de cámara para esta aplicación. El de internet es usado para enviar un email vía API Java Mail como se verá más adelante, y el de cámara para modificar la imagen de perfil del usuario.

- El tema elegido como podremos ver en las figuras de cada pantalla es en azules y amarillos.

- Por último, queda destacar que en todas las activities se ha sobrescrito el método onBackPressed(). Éste, da la función al botón de atrás de nuestros dispositivos. Para la aplicación el botón de atrás tiene la funcionalidad de salir de la aplicación en todos los casos, en ninguno volverá a la activity anterior, ya que para el movimiento entre activities está el menú desplegable o los botones de la propia aplicación.

5.2. DIAGRAMA DE ACTIVIDADES

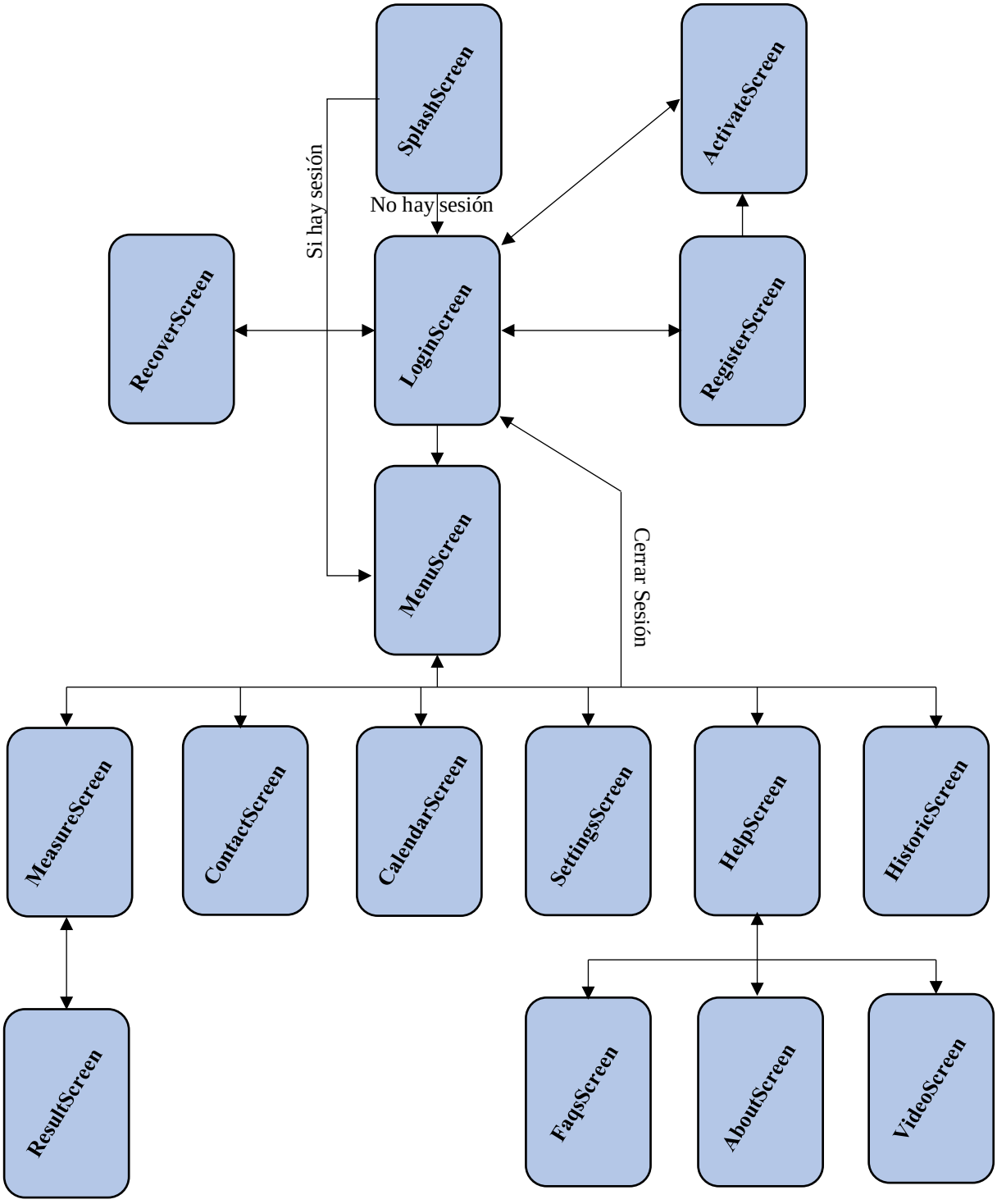


FIG. 12. DIAGRAMA DE ACTIVIDADES

5.3. ACTIVIDADES

5.3.1. SplashScreen

En la Fig.13. se muestra como es la primera pantalla de la aplicación. Esta pantalla se denomina SplashScreen, y suele ser utilizada a modo de presentación de la empresa desarrolladora, para mostrar el logo de aplicación, etc. Cualquier tarea de introducción.

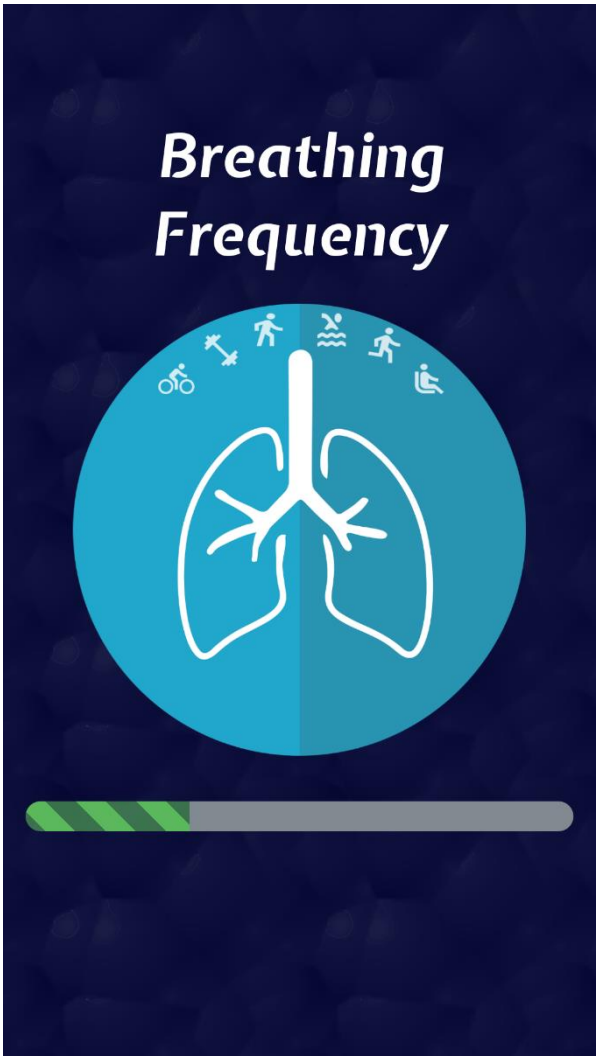


FIG. 13. SPLASHSCREEN

La funcionalidad de esta activity es la siguiente:

- Se trata de la actividad launcher, es decir, será siempre la aplicación que se muestre primero al abrir.

- Carga las librerías de la aplicación.

- Comprueba el idioma que está seleccionado en la aplicación, y si hay alguna sesión iniciada. De esta manera no hace falta logarse cada vez que se entre en la aplicación.

- En función de si hay sesión o no iniciada, redirige a la pantalla de inicio de sesión o a la pantalla del menú.

- Va actualizando la ProgressBar simulando la carga de la aplicación, en este caso durará cuatro segundos.

Elementos incluidos:

- TextView para el título.
- ImageView para el logo.
- ProgressBar para la barra de progreso.

5.3.2. LoginScreen

En la Fig.14. se muestra la pantalla de iniciar sesión. Esta pantalla se utiliza para acceder a la aplicación con tu cuenta de usuario.

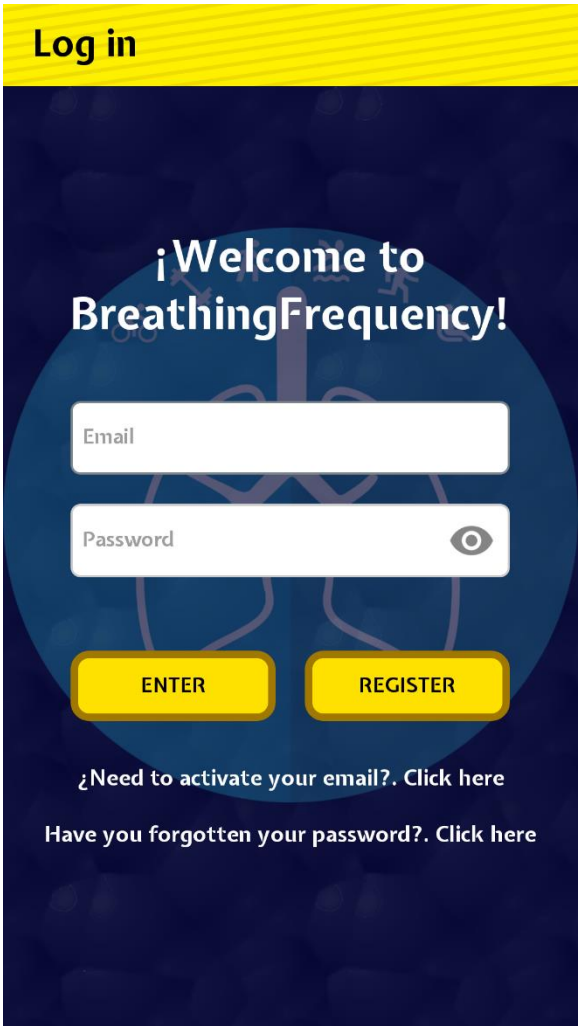


FIG. 14. LOGINSCREEN

La funcionalidad de esta activity es la siguiente:

- En primer lugar, pide al usuario que conceda los permisos necesarios para utilizar la aplicación. En caso de denegarlos, se cierra la aplicación.

- Si hacemos click en el botón "Enter", comprueba que el email está en BD, en caso de que no, muestra un Toast informando. Si está en BD, comprueba que la contraseña es correcta, luego comprueba si el email está activo, y si todo está bien te redirige a la pantalla de menú. Si la contraseña no fuera correcta muestra un Toast informando. En el caso de un inicio correcto, además, graba en tanto en BD como en las sharedpreferences, un nuevo token de sesión.

- Si hacemos click en el botón "Register", te redirige a la pantalla de registro.

- Si hacemos click en el texto de activar el email, nos pide que introduzcamos qué email queremos activar, y a continuación comprueba en BD si está activado o no. Si ya estuviera activado nos los indica mediante un Toast, en caso contrario nos envía una notificación con un código de activación [21] y nos redirige a la pantalla de activación.

- Si hacemos click en el texto de recuperar la contraseña, no pide que introduzcamos el email para el cual queremos recuperar la contraseña. A continuación, nos pregunta mediante un Dialog si ya tenemos un código de recuperación de contraseña. En caso negativo, manda un email automático a la cuenta de usuario con el código, y nos redirige a la pantalla de recuperar contraseña. En caso de que ya tuviéramos un código se salta el paso de enviar el email y directamente redirige a la pantalla de recuperar contraseña. Para esta funcionalidad he utilizado la librería JavaMail [22].

- Si pulsamos sobre el ImageView del ojo, la contraseña se mostrará visible, y si volvemos a pulsar se mostrará oculta.

Elementos utilizados:

- LinearLayout con un TextView para simular la barra de navegación superior.

- TextView para el título.

- Dos EditText para introducir el email y la contraseña

- Dos Button para los botones de acción.

- Dos TextView para los textos de recuperar contraseña y activar email, ambos con listener[6] para responder ante un evento click.

- Dos ImageView, uno para el logo en forma de marca de agua del fondo, y otro para el ojo del EditText.

5.3.3. RegisterScreen

En la Fig.15. se muestra la pantalla de registro. Esta pantalla se utiliza para registrar nuevos usuarios de la aplicación por primera vez.

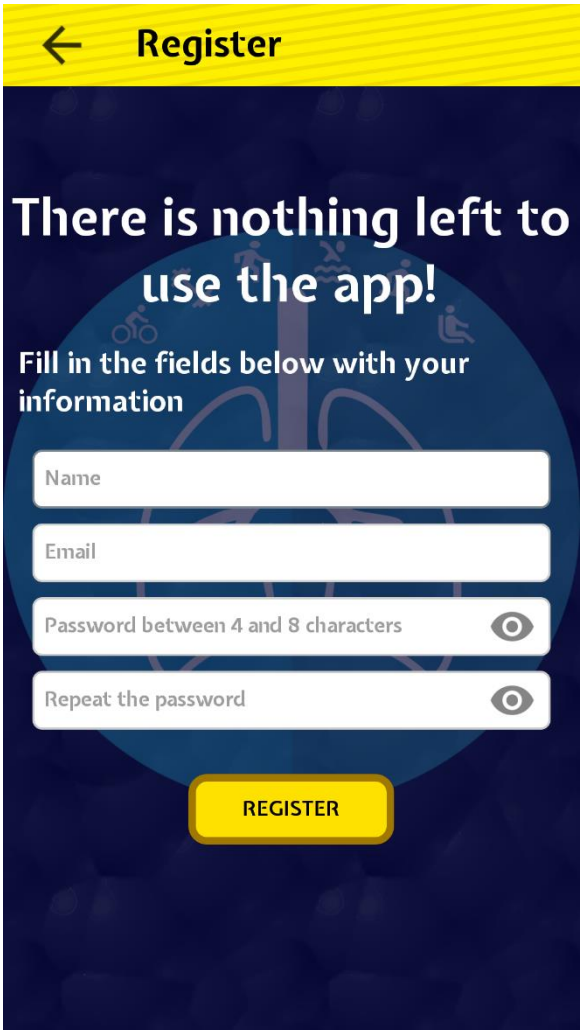


FIG. 15. REGISTERSCREEN

La funcionalidad de esta de esta activity es la siguiente:

- Si se pulsa la flecha contenida en la Toolbar te redirige a la pantalla de inicio de sesión.

- Al pulsar el botón "Register", comprueba primero si se han rellenado todos los campos, en caso negativo informa mediante un Toast. A continuación, comprueba que el email tenga el formato apropiado, en caso contrario informa mediante un Toast.

- Finalmente, si el punto anterior es correcto, comprueba que el email no esté registrado con anterioridad, si es así informa con un Toast, si no, introduce el nuevo usuario en BD. En este momento el flag de activo del cliente está a 0, lo que indica que no está activado.

- De manera complementaria los dos ojos que aparecen sobre los EditText de contraseñas, al pulsarlos muestra la contraseña de manera visible, y si volvemos a pulsarlo, de nuevo la ocultan.

- En el momento de pinchar por primera vez el EditText del email se muestra un Dialog informando al usuario de que puede usar un email inventado, como dije en las consideraciones de diseño.

Elementos utilizados

- LinearLayout con un TextView y un Button para simular la Toolbar superior.

- Dos TextView para el título y el subtítulo.

- Tres ImageView, dos para los ojos que muestran las contraseñas y una para el logo en forma de marca de agua del fondo.

- Cuatro EditText para introducir el nombre, email, contraseña y repetir la contraseña.

5.3.4. ActivateScreen

En la Fig.16. se muestra la pantalla de activación de email. Esta pantalla se utiliza para estar seguros de que le registro lo ha llevado a cabo una persona, solicitándole un código que se le manda por notificación.

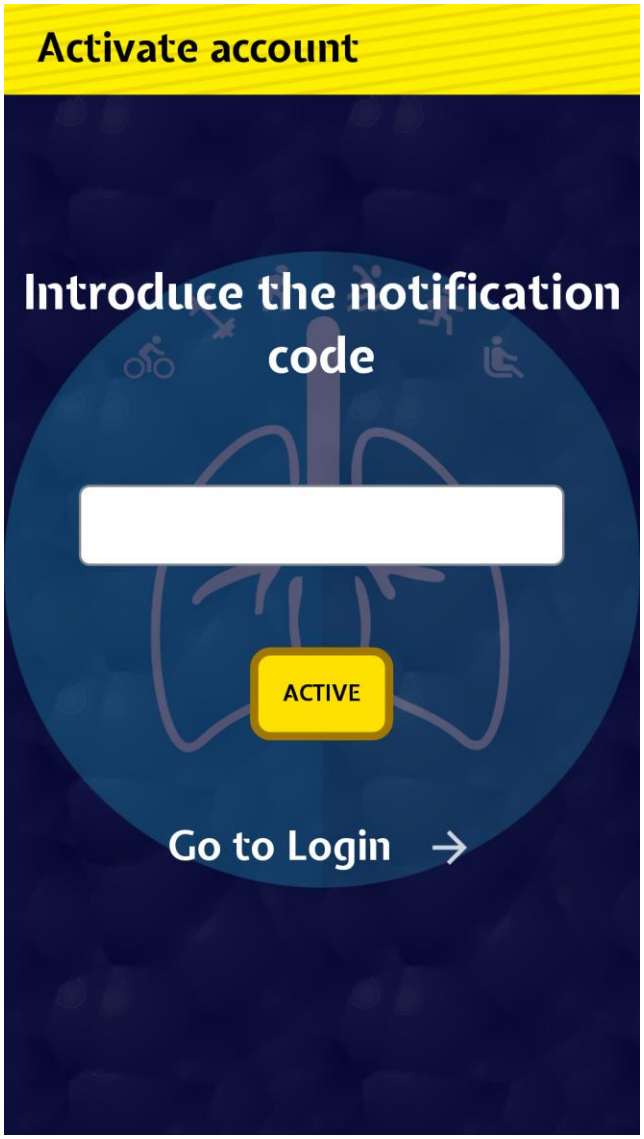


FIG. 16. ACTIVATESCREEN

La funcionalidad de esta activity es la siguiente:

- Para llegar a esta pantalla se ha tenido que venir desde la pantalla de Inicio de sesión, o de la de registro. En ambas ocasiones ha tenido que llegar una notificación a la barra de estado del dispositivo indicando un código de cinco dígitos alfanuméricos. Este código debe introducir en el EditText que aparece.

- En el momento de pulsar el botón "Active", comprueba que el código introducido es igual al código de la notificación, en caso contrario lo indica con un Toast. Si es correcto modifica el flag de activo del cliente en BD poniéndolo a 1, lo que indica que está activado. Por último, no redirige a la pantalla de inicio de sesión

- Si pulsamos en el TextView que pone ir a login, nos redirige a la pantalla de inicio de sesión.

Elementos utilizados

- LinearLayout con TextView para simular la Toolbar superior.
- Dos TextView para el título, y para volver a la pantalla de inicio.
- Un Button para realizar la lógica de activación de un usuario.

5.3.5. RecoverScreen

En la Fig.17. se muestra la pantalla de recuperar contraseña. Esta pantalla se utiliza para los casos que no recuerdan la contraseña y quieren reestablecerla.

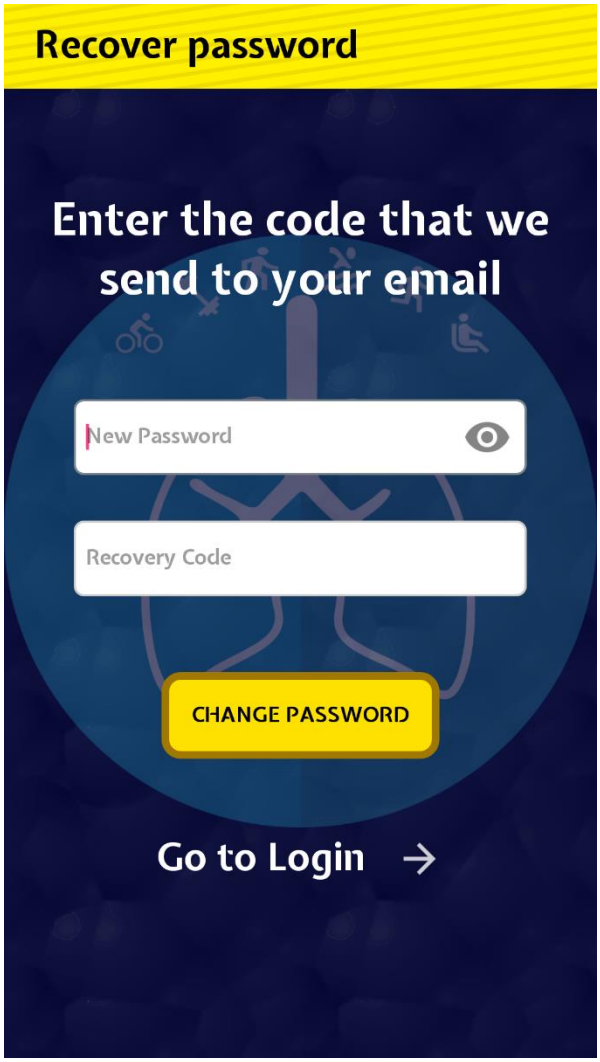


FIG. 18. RECOVERSCREEN

La funcionalidad de esta activity es la siguiente:

- Para llegar a esta pantalla, lo hemos tenido que hacer desde la pantalla de inicio de sesión. Antes de redirigirnos nos habrá preguntado si tenemos un mensaje de recuperación de contraseña, como el que se muestra en la Fig.18.

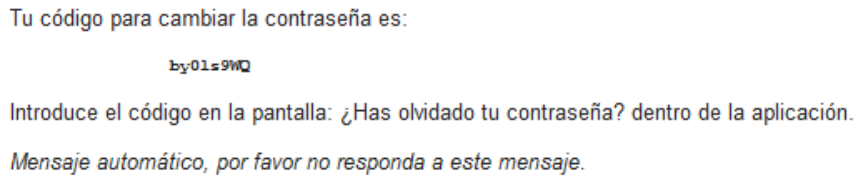


FIG. 17. EJEMPLO DE EMAIL

- Al pulsar en el botón “Change Password”, comprueba que el código que has introducido corresponde al que está en BD almacenado como código de recuperación de contraseña. En caso negativo, informa mediante un Toast. En caso afirmativo, guarda la nueva contraseña y borra el código de recuperación que tenías en BD. Finalmente, redirige a la pantalla de inicio de sesión.

- Si pulsamos en el TextView que pone ir a login, nos redirige a la pantalla de inicio de sesión.

- Del mismo modo que en la pantalla de iniciar sesión, o la de registro, el ImageView del ojo muestra y oculta la contraseña.

Elementos utilizados

- LinearLayout con TextView para simular la Toolbar superior.

- Dos TextView, uno para el título y otro para el texto de ir a la pantalla de iniciar sesión.

- Un Button para la lógica de recuperar contraseña.

- Dos ImageView, uno para el logo como marca de agua del fondo, y otro para el ojo del mostrar/ocultar la contraseña.

- Dos EditText para introducir la nueva contraseña y el código de recuperar contraseña recibido por email.

5.3.6. MenuScreen

En la Fig.19. se muestra la pantalla de menú. Esta pantalla se utiliza para poder navegar a las diferentes pantallas de la aplicación que nos ofrecen a su vez toda la funcionalidad de la aplicación.

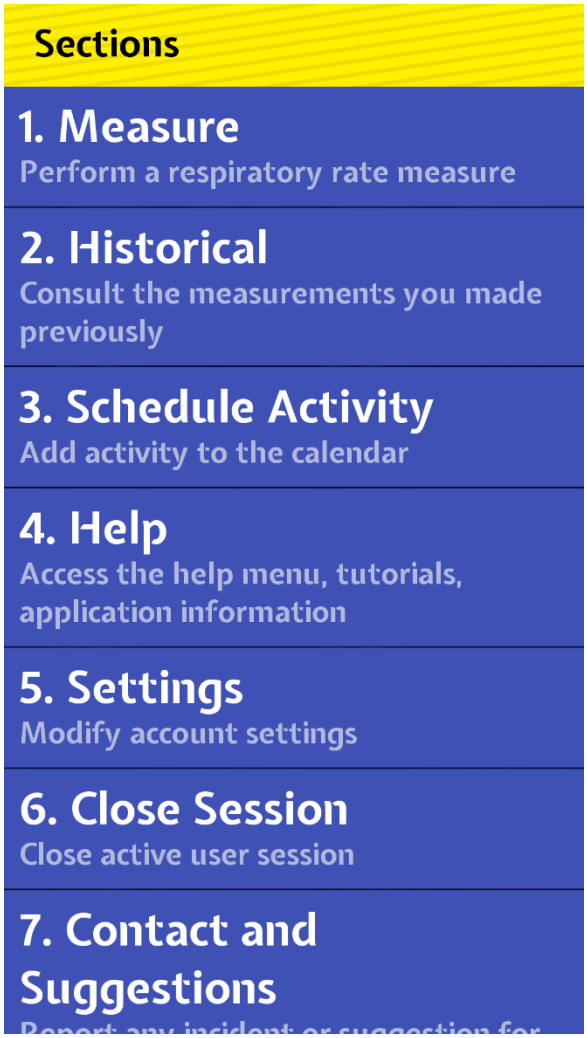


FIG. 19. MENUSCREEN

La funcionalidad de esta activity es la siguiente:

- Dependiendo de la opción del ListView en la que pulsemos nos redirige a una pantalla o a otra.

- En caso de pulsar en "Cerrar Sesión", nos redirige a la pantalla de iniciar sesión, y borra el token de sesión tanto de las sharedpreferences, como de la BD.

- En este caso el dispositivo utilizado tiene la pantalla lo suficientemente grande para abarcar todas las opciones del menú, pero en caso de que no fuera así, el ListView esta integrado en un ScrollView, y de esta manera se podría deslizar la pantalla para ver todas las opciones.

Elementos utilizados:

- LinearLayout con TextView para simular la Toolbar superior.

- ListView que contiene las diferentes pantallas de la aplicación. Para diseñar las diferentes casillas del ListView, se han utilizado dos clases auxiliares. Una con un constructor que contiene dos Strings. Y un adaptador que las dispone de la forma que se ve en la Fig.19.

5.3.7. SettingsScreen

En la Fig.20. se muestra la pantalla de ajustes. Esta pantalla se utiliza para modificar los datos personales de la cuenta de usuario, y para eliminar la cuenta.

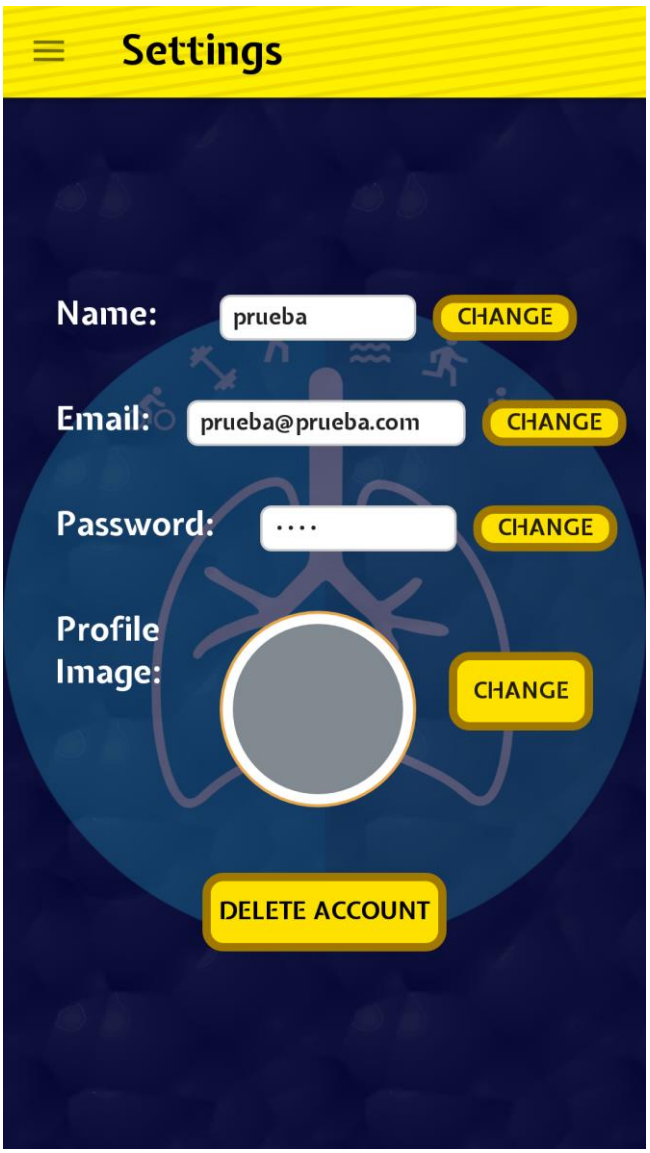


FIG. 20. SETTINGSSCREEN

La funcionalidad de esta activity es la siguiente:

- Cuando accedemos a esta pantalla nos muestra los datos actuales que tenemos guardados en BD. Si pulsamos en cualquiera de los botones "Change" se abre un Dialog que nos pide el nuevo valor del dato que queramos cambiar, y una vez aceptemos se graba en BD.

- Para el caso de cambiar la imagen de perfil, se nos abrirá la cámara frontal para hacernos una foto [23]. Cuando aceptemos una foto, aparecerá en el Thumbnail circular, y se guardará en BD. Para saber cómo guardar imágenes en SQLite me he apoyado en una entrada del foro StackOverFlow [24].

- Si pulsamos en el botón de borrar cuenta, nos pregunta mediante un Dialog si estamos seguros, y si aceptamos, eliminará primero todas las imágenes de históricos que tuviéramos guardadas, y a continuación nuestra información de usuario.

- Para las funciones de BD de esta pantalla he tenido que agregarles un segundo de espera después de acabar porque si el usuario cambiaba varios datos de manera rápida, sólo se guardaba correctamente el primero de ellos.

- Si pulsamos en el Button de la Toolbar, se abre el menú desplegable de navegación.

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.
- Cuatro TextView para los títulos de los datos.
- Tres EditText para los tres datos que son texto.
- Un Thumbnail circular para mostrar la imagen de perfil.
- Cinco Button para las diferentes lógicas de la pantalla.
- Un ImageView para el logo en forma de marca de agua del fondo.

5.3.8. ContactScreen

En la Fig.21. se muestra la pantalla de contacto y sugerencias. Esta pantalla se utiliza para reportar incidencias detectadas por los usuarios, realizar consultas al desarrollador, o para hacer sugerencias de mejora sobre la aplicación.

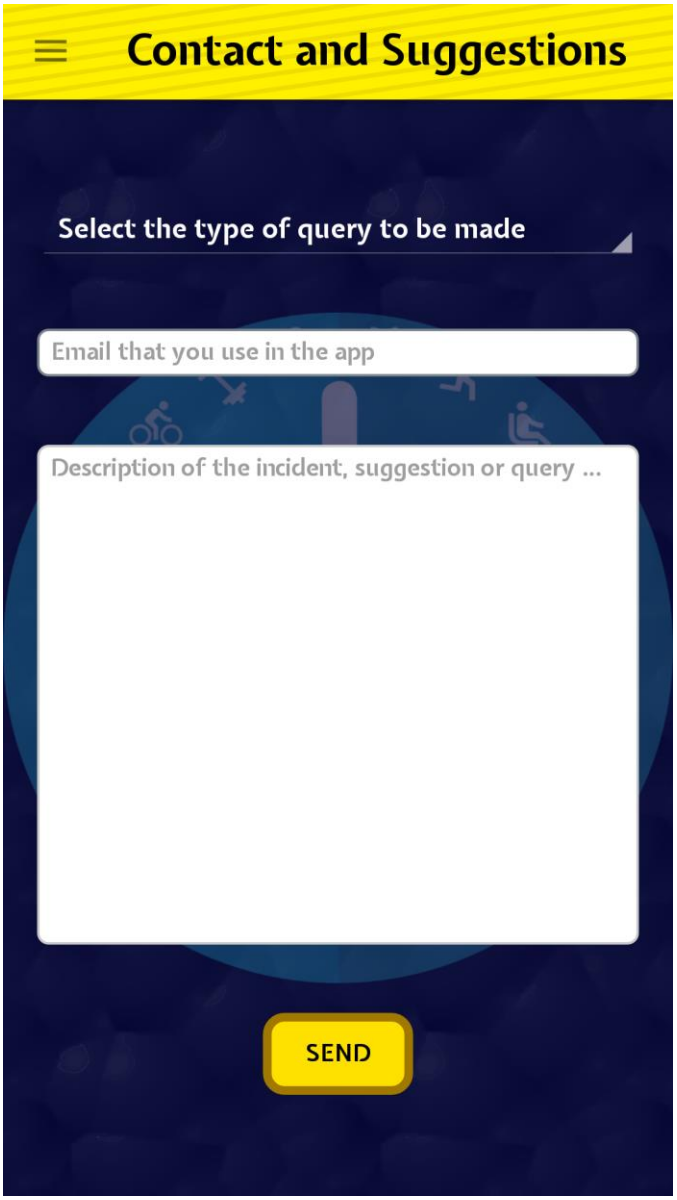


FIG. 21. CONTACTSCREEN

La funcionalidad de esta activity es la siguiente:

- Si pulsamos en el botón de enviar, comprueba que hayas elegido un tipo de consulta en el spinner. En caso negativo muestra un Toast. Comprueba que hayas introducido el email que utilizas en la aplicación. Y por último comprueba que hayas escrito una descripción de la consulta. Si todas las comprobaciones son correctas, te mostrará todos los clientes de correo instalados en tu dispositivo, y tendrás que elegir el que quieras para enviar la consulta vía email. Ésta, es otra forma diferente de enviar un email sin usar el API de JavaMail [25].

- Si pulsamos en el Button de la Toolbar, se abre el menú desplegable de navegación.

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.
- Spinner para la lista desplegable de tipos de consulta.
- Dos EditText para introducir el emaily la descripción de la consulta.
- Un Button para la lógica de la pantalla.

5.3.9. CalendarScreen

En la Fig.22. se muestra la pantalla de programar actividad. Esta pantalla se utiliza para añadir al calendario del dispositivo un evento/actividad física que vayamos a realizar y junto a ella vayamos a tomarnos la frecuencia respiratoria.

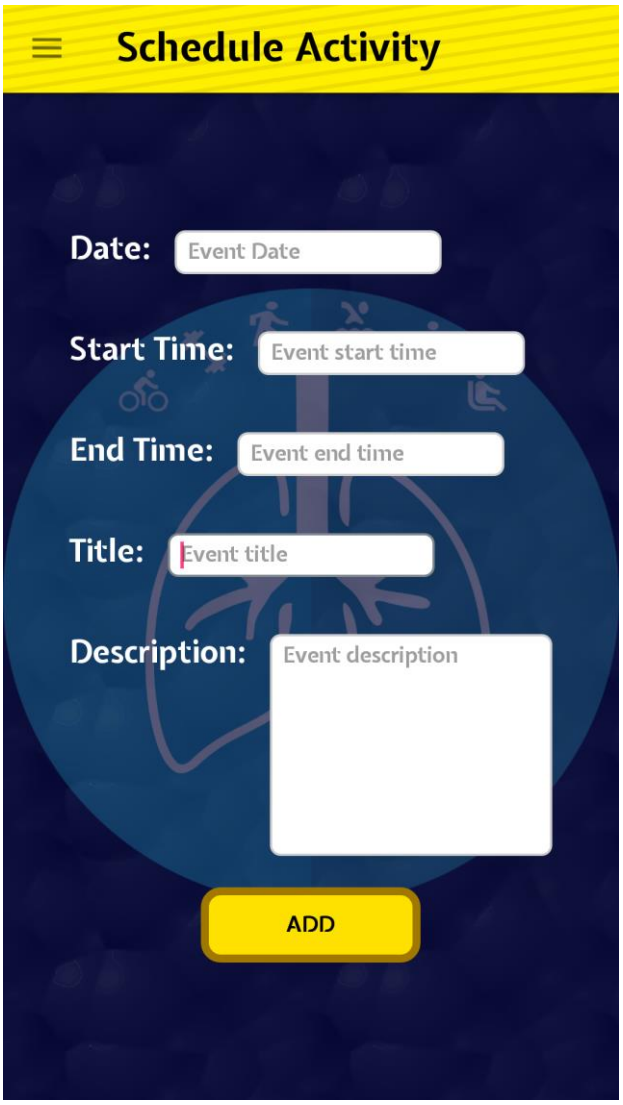


FIG. 22. CALENDARSCREEN

La funcionalidad de esta activity es la siguiente:

- Al pulsa el botón añadir, comprueba que todos los campos se hayan rellenado, si no es así muestra un Toast. Si están todos rellenos, comprueba que la fecha del evento no sea anterior a hoy, y que la hora de inicio no sea posterior a la de fin, si alguno de los casos anteriores se diera muestra un Toast. Si todo fuera correcto, abre el calendario del dispositivo en la pantalla de añadir evento con los campos cumplimentados [26] [27], a falta de darle a añadir.

- Para rellenar la fecha del evento he utilizado un DatePicker como el de la Fig.23. [28], y para rellenar las horas de inicio y fin, he utilizado un TimePicker como el de la Fig.24.

- Si pulsamos en el Button de la Toolbar, se abre el menú desplegable de navegación.

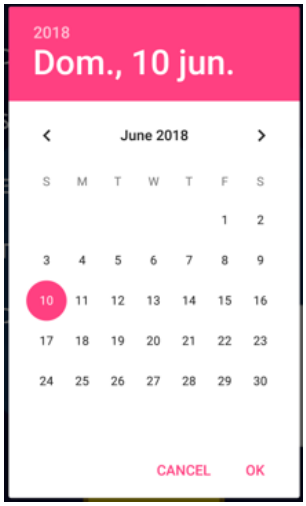


FIG. 24. DATEPICKER

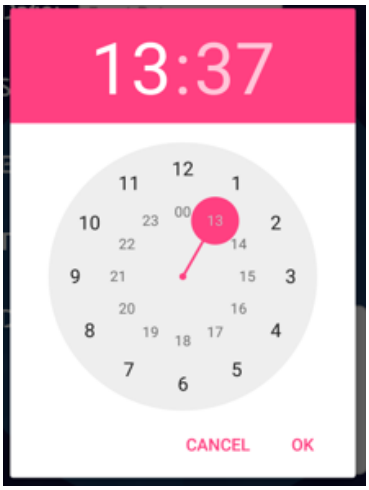


FIG. 23. TIMEPICKER

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.
- Cinco TextView para los textos de los datos del evento.
- Cinco EditText para introducir los datos del evento.
- Un Button para la lógica de la pantalla.
- Un ImageView para el logo en forma de marca de agua del fondo.
- Un DatePicker para rellenar la fecha del evento.
- Un TimePicker para rellenar las horas de inicio y fin del evento.

5.3.10. HistoricScreen

En la Fig.25. se muestra la pantalla de las medidas históricas cuando aún no se ha guardado ninguna, y en la Fig.26 se muestra la misma pantalla cuando hay imágenes guardadas. Esta pantalla se utiliza para mostrar las imágenes que se hayan ido guardando con anterioridad a modo de histórico.

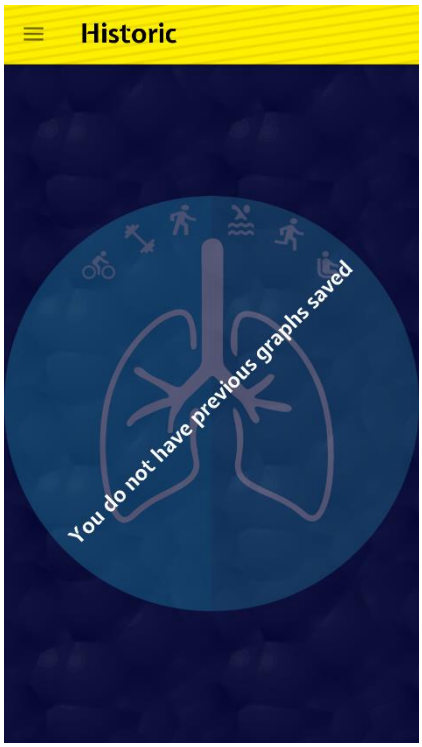


FIG. 25. EMPTYHISTORIC

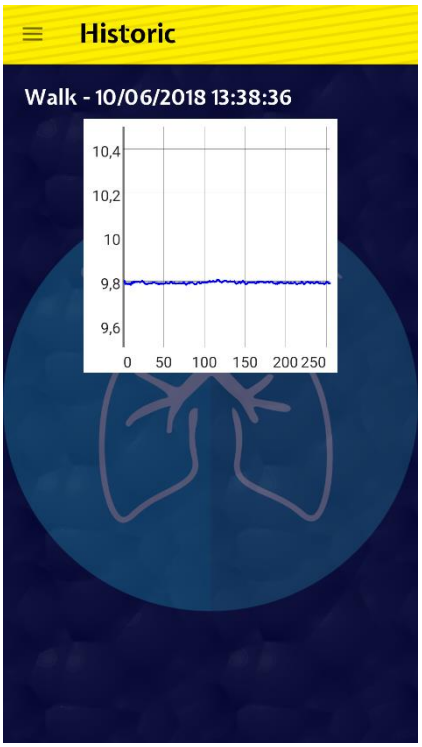


FIG. 26. HISTORICSCREEN

Esta pantalla no tiene funcionalidad real. Nada más cuando se accede a ella busca en BD si hay alguna imagen guardada y las muestra una debajo de otra. Esta pantalla está contenida en un ScrollView para poder ver las cinco imágenes guardadas deslizando la pantalla.

Si pulsamos en el Button de la Toolbar, se abre el menú desplegable de navegación.

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.
- Un TextView por cada título de cada imagen
- Un ImageView para cada imagen guardada.
- Un Image View para el logo del fondo como marca de agua.

5.3.11. MeasureScreen

En la Fig.27. se muestra la pantalla medir. En la Fig.28. se muestra la misma pantalla tras haber pulsado Iniciar. Esta pantalla se utiliza para tomar valores del acelerómetro a modo de medida de frecuencia respiratoria.

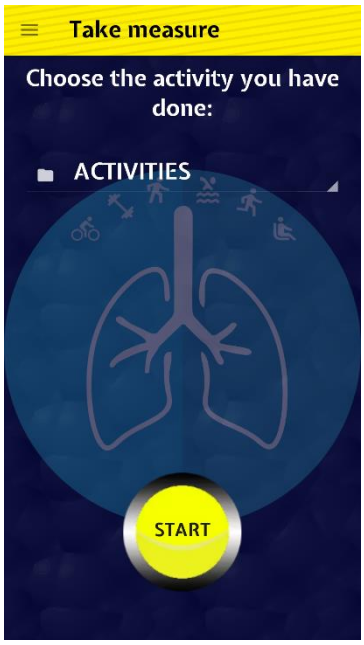


FIG. 27. MEASURESCREEN

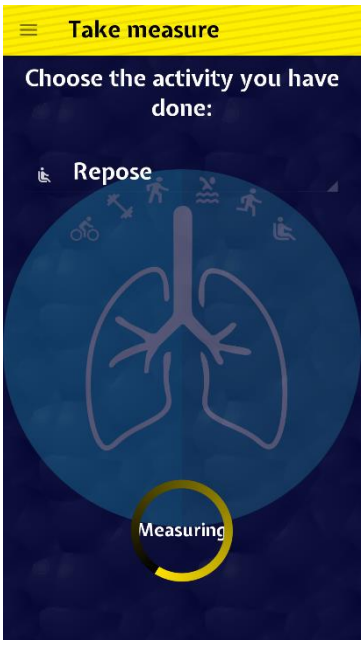


FIG. 28. MEASURING

La funcionalidad de esta activity es la siguiente:

- Cuando pulsamos en el botón Iniciar, comprueba que se haya seleccionado una actividad en el Spinner. En caso negativo, muestra un Toast. En caso afirmativo pasa a la Fig.28. y durante 15 segundos estará tomando valores del sensor acelerómetro. Cuando los 15 segundos pasen, nos redirige a la pantalla de resultado.

- Si pulsamos en el Button de la Toolbar, se abre el menú desplegable de navegación.

- En este caso, he tenido que usar programación multihilo para los 15 segundos de medida, ya que si no se bloqueaba la interfaz de usuario. He utilizado una clase auxiliar heredada de AsynkTask [29] para esta labor.

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar.

- Dos TextView, uno para el título y otro para el mensaje de “Midiendo…”.
- Un Spinner personalizado con imagen y texto para las actividades.

- Un Button para la lógica de la pantalla.

- Una ProgressBar circular para hacer ver que está operando en segundo plano realizando la medida.

5.3.12. ResultScreen

En la Fig.29. se muestra la pantalla del resultado de la medición. Esta pantalla se utiliza para mostrar el resultado de la medición después de hacer toda la lógica interna para, a partir de los valores que llegan de la pantalla anterior, saber cuantas respiraciones se han hecho.

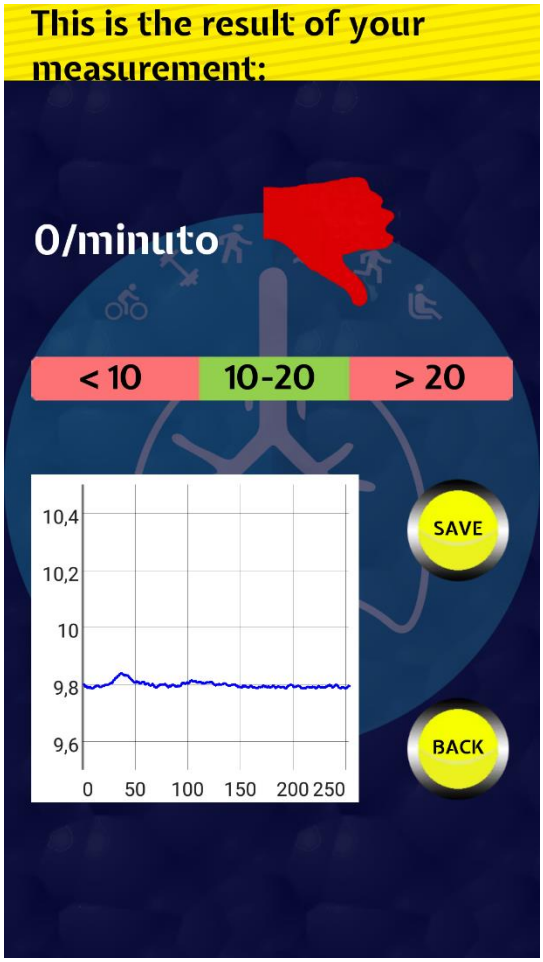


FIG. 29. RESULTSCREEN

La funcionalidad de esta activity es la siguiente:

- Desde la pantalla de medir, nos llega un grupo de valores del acelerómetro. Con ellos se hacen unas operaciones para intentar llegar a obtener cuantas respiraciones se han hecho. Además, se pintan los valores en una gráfica. Para esto último he utilizado el API open source de GraphView [30].

- Si pulsamos el botón de guardar, se comprueba si ya hay cinco imágenes guardadas para ese usuario. En caso afirmativo, se elimina la más antigua y se guarda ésta en BD. En caso negativo, se guarda el gráfico en BD.

- Si pulsamos el botón volver, nos redirige a la pantalla de medir.

- El standard mostrado en la barra de colores varía en función de la actividad escogida en la pantalla de medir.

Elementos utilizados

- ListView con TextView para simular la Toolbar superior.

- Cuatro TextView, uno para mostrar el número de respiraciones y tres para mostrar el standard.

- Tres ImageView, una para mostrar la mano que indica si están dentro del standard, otra para la barra donde se muestran el standard y otra para el logo de fondo a modo de marca de agua.

- Un GraphView para mostrar los valores pintados como una gráfica.

- Dos Buttons para la lógica de la pantalla.

5.3.13. HelpScreen

En la Fig.30. se muestra la pantalla de ayuda. Esta pantalla se utiliza para mostrar la información sobre la aplicación, los videotutoriales de uso y las preguntas frecuentes.

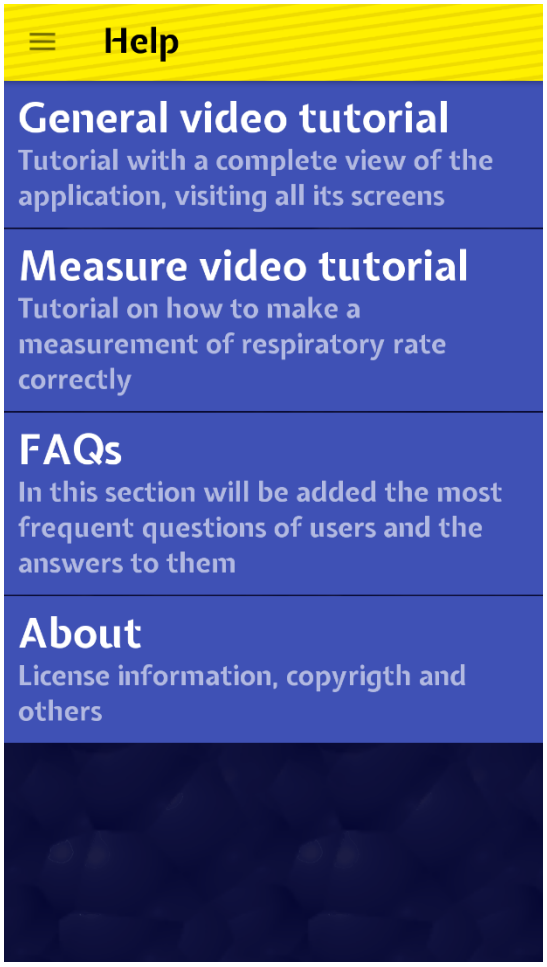


FIG. 30. RESULTSCREEN

La funcionalidad de esta activity es la siguiente:

- Dependiendo de la opción del ListView que se pulse nos redirige a una pantalla nueva o a otra.

- Si pulsamos en el Button de la Toolbar, se abre el menú desplegable de navegación.

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.

- ListView para mostrar las diferentes opciones. Para este ListView se han reutilizado las dos clases auxiliares que expliqué en el ListView de la pantalla del menú.

5.3.14. AboutScreen

En la Fig.31. se muestra la pantalla de acerca de. Esta pantalla se utiliza para mostrar información sobre las licencias y proyectos que se hayan utilizado en la aplicación.

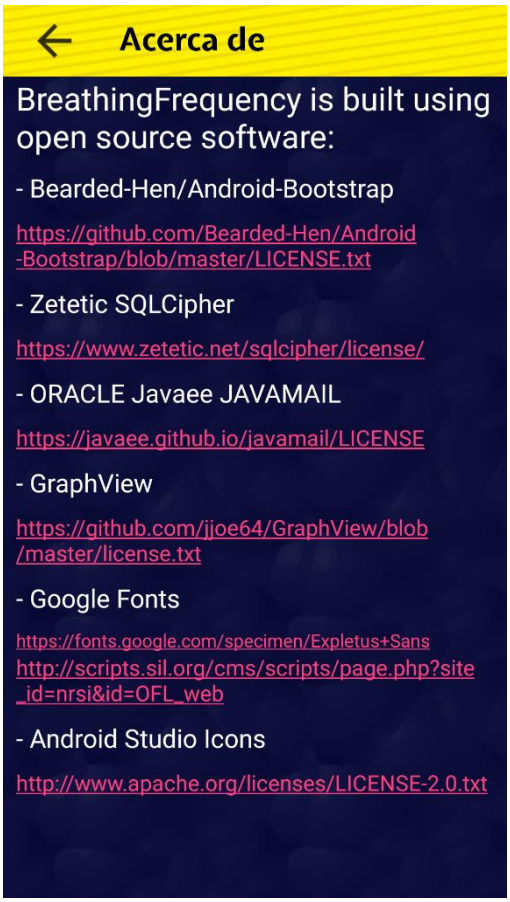


FIG. 31. ABOUTSCREEN

Esta pantalla no tiene funcionalidad realmente, únicamente muestra los textos predefinidos tal cual se ven en la Fig.31. Si pulsamos en las diferentes urls que hay nos redirigirá al navegador predeterminado del dispositivo y nos mostrará las diferentes licencias de los proyectos utilizados.

Si pulsamos en el Button de la Toolbar, nos redirige a la pantalla de ayuda.

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.
- Tantos TextViews como textos se ven en la Fig.31.

5.3.15. FaqScreen

En la Fig.32. se muestra la pantalla de Faqs o preguntas frecuentes. Esta pantalla se utiliza para mostrar las preguntas más frecuentes que lleguen a través del formulario de contacto que vimos en ContactScreen, y de esta manera intentar reducir el número de consultas recibidas.

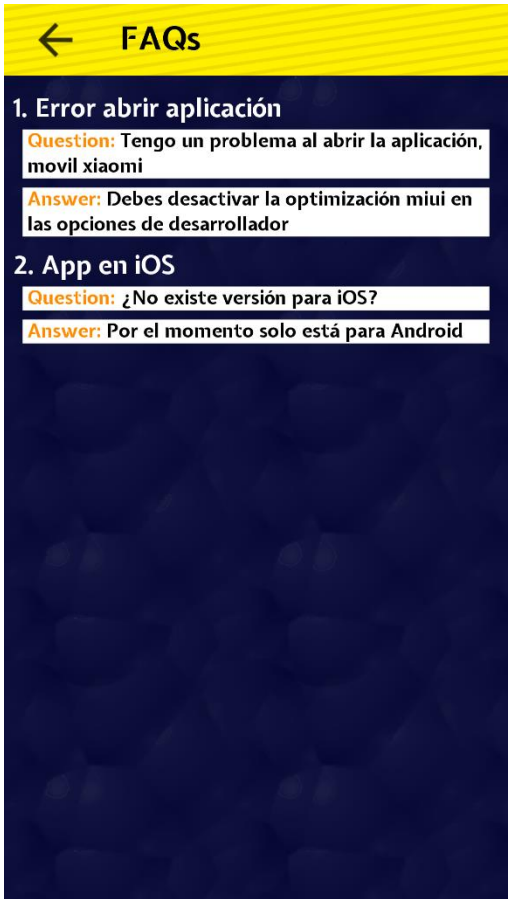


FIG. 32. FAQSCREEN

Esta pantalla realmente no tiene funcionalidad en sí. Simplemente hace una consulta a BD en busca de todas las faqs que haya guardadas y las muestra con el formato que vemos en la Fig.32.

En este caso cada una de las Faqs hay que meterlas a mano en la BD mediante una función que hay creada para ello en el código.

Si pulsamos en el Button de la Toolbar, nos redirige a la pantalla de ayuda.

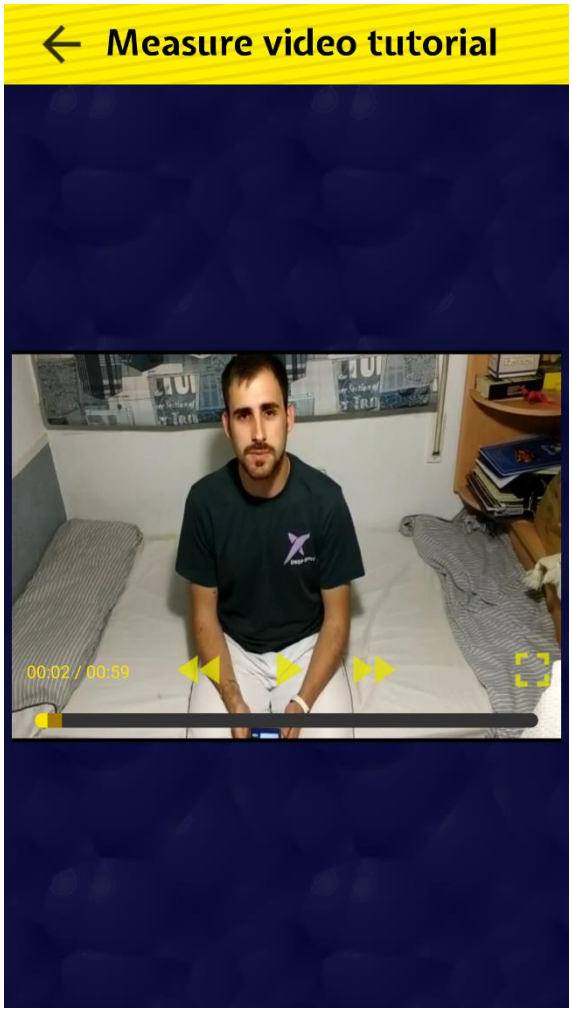
Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.

- Tantos TextView como Faqs haya en BD. En este caso podemos ver que cada Faq consta de tres TextView, uno para el título, otro para la pregunta y otro para la respuesta.

5.3.16. VideoScreen

En la Fig.33. se muestra la pantalla de videotutoriales. Esta pantalla se ha utilizado para mostrar los dos videotutoriales que hay grabados para la aplicación. Uno enseña a utilizar la aplicación y otro muestra como tomar una medida de manera correcta. Ambos están grabados en inglés de cara a que van a ser reproducidos a nivel internacional.



La funcionalidad de esta activity es la siguiente:

- Al cargarse, depende de qué video se haya escogido en la pantalla de ayuda, se carga un video u otro en el SurfaceView asociándose a un objeto de tipo MediaPlayer.

- Al pulsar en la superficie del SurfaceView se muestra durante 4 segundos el MediaController personalizado, con el que se puede interactuar para inciar el video, pausarlo, avanazarlo, rebobinarlo, o pasar a pantalla completa.

- Si pulsamos en el Button de la Toolbar, nos redirige a la pantalla de ayuda.

FIG. 33. VIDEOSCREEN

Elementos utilizados

- LinearLayout con TextView y Button para simular la Toolbar superior.
- SurfaceView para mostrar el video sobre él.
- LinearLayout para montar el MediaController personalizado tal y como se ve en la Fig.33.

En esta pantalla me voy a detener para explicar de manera detallada el funcionamiento del MediaController y como cambiar a pantalla completa, puesto que no es algo sencillo. Para ello me he apoyado en tres tutoriales [31] [32].

En primer lugar, hay que tener claro el ciclo de vida de un MediaPlayer. En la Fig.34. podemos ver cómo funciona.

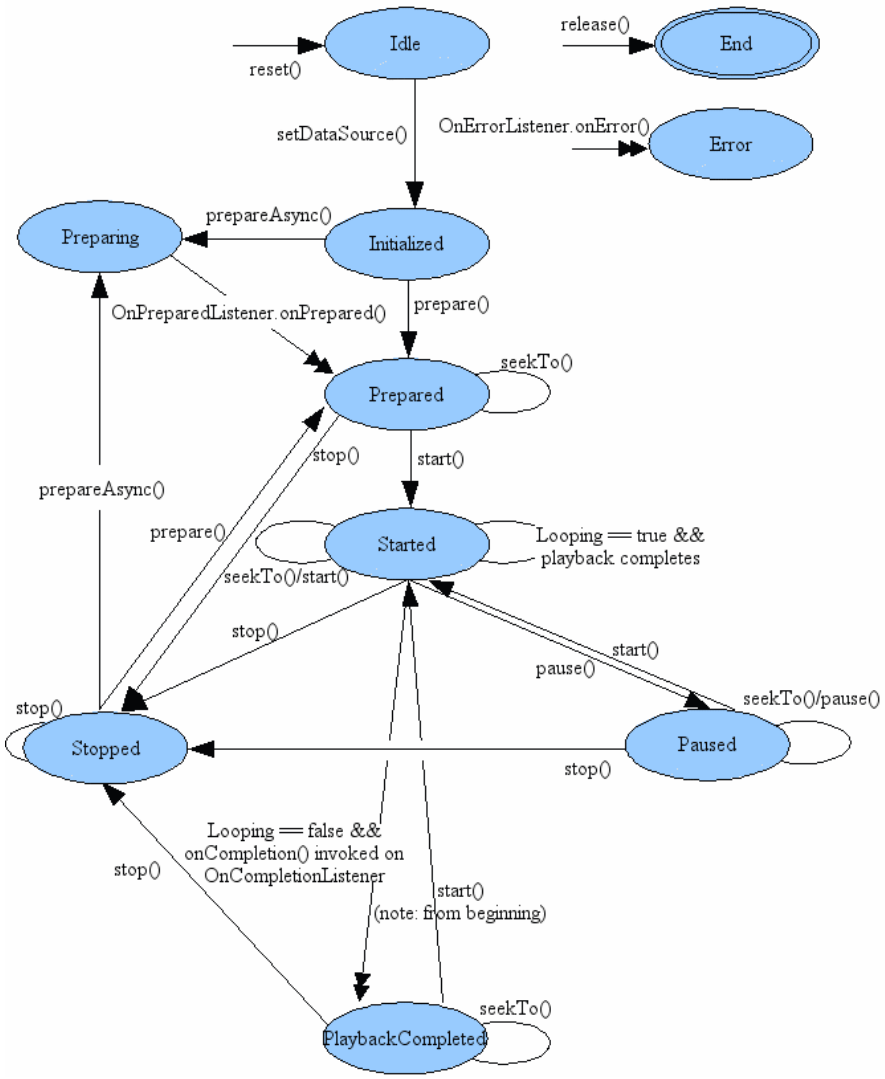


FIG. 34. MEDIAPLAYER ***Fuente: Android Developers [33]***

Los estados que nos interesan son Initialized y Prepared. Cuando creamos el MediaPlayer y lo asociamos a un video (en este caso), pasa al estado Initialized. En este estado NO se puede hacer start() al MediaPlayer. Para ello hay que ejecutar antes prepare() sobre él. En ese momento ya estará en Prepared y listo para poder utilizarse.

Ahora conviene hablar de las funciones de la SeekBar. Las más importantes son:

- getCurrentPosition(): obtiene el progreso actual de la barra, que si previamente la hemos asociado a la duración del video, corresponderá al segundo del video actual.
- onStartTrackingTouch(): esta función se llama cuando hacemos click sobre la SeekBar.

- onStopTrackingTouch(): esta función se llama cuando se deja de pulsar la SeekBar.
- onCompletion(): esta función se llama cuando acaba la reproducción.

Así pues, teniendo estas funciones en mente, y como he dicho habiendo asociado la duración del MediaPlayer al progreso de la SeekBar, ya somos capaces de avanzar y retroceder el video con los botones del MediaController.

El play y el pause tan tienen como funcionalidad poner a correr el MediaPlayer y paudarlo respectivamente. Para esto solo hay que tener en cuenta la Fig.34.

Por último, me queda explicar el cómo pasar el vídeo a pantalla completa. En el caso del videotutorial de uso de la aplicación, no tiene mucho misterio. Tan solo eliminamos la Toolbar de la vista y ampliamos el tamaño del vídeo para que coincida con el tamaño de la pantalla. Esto lo puedo hacer así porque está grabado en vertical.

Sin embargo, para el videotutorial de como tomar una medida de manera correcta, puesto que se ha grabado en horizontal, el paso a pantalla completa es más complejo. La línea de código que cambia la orientación de la pantalla es la siguiente:

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

Cuando modificamos la orientación de la pantalla se llama al método onSaveInstanceState(Bundle bundle), y se vuelve a crear el activity. Debido a esto todos los elementos y progreso se pierde. De esta forma en el método mencionado tendremos que guardar el progreso actual del video para continuarlo por donde estaba, la url del video que se estaba reproduciendo, y todo lo que necesitemos para poder crear de nuevo el activity[2] en orientación horizontal. Del mismo modo cuando pasemos de horizontal a vertical. Por último para que todo lo anterior funcione, en el método onCreate() del activity, recordemos que es en el cual se crea todo, tendremos que comprobar si el Bundle pasado por el método onSaveInstanceState(Bundle bundle) tiene datos. En caso de que sí, sabemos que viene de un cambio de orientación. Además, comprobaremos la orientación actual para crear el activity de una manera o de otra.

Con todo esto en mente seremos capaces de reproducir lo que yo he montado para esta pantalla.

5.3.17. Menú Desplegable

En la Fig.35. se muestra el menú desplegable del que hemos estado hablando en algunas de las pantallas [34]. Éste, tiene la función de facilitar la navegación entre las distintas pantallas de la aplicación.

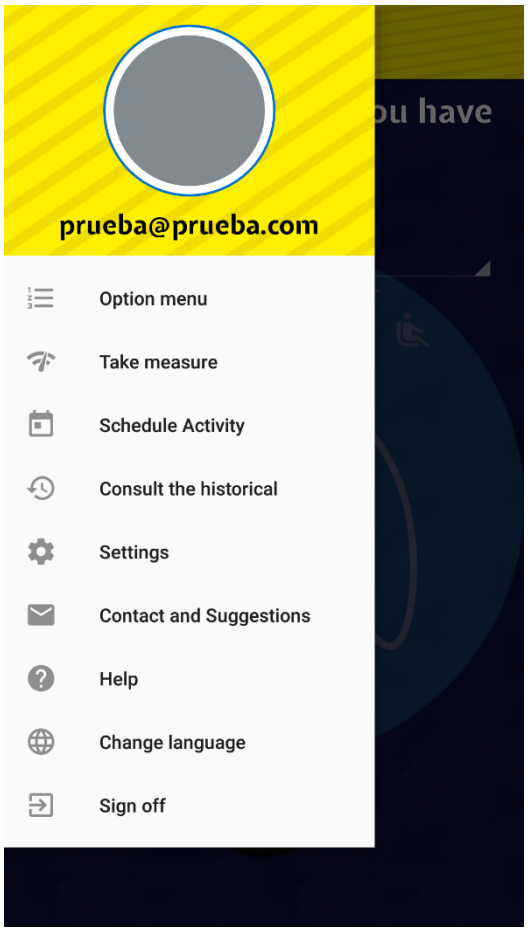


FIG. 35. MENÚ DESPLEGABLE

La funcionalidad que tiene este menú es redirigir al usuario a las pantallas según que opción elija.

Además, muestra la imagen de perfil y el email del usuario a modo de cabecera.

En este menú podemos observar una nueva funcionalidad no presente en ninguna actividad. La de cambiar el idioma de la aplicación.

Si se elige la opción de cambiar idioma [35], se muestra un Dialog como el de la Fig.36. y tras elegir un idioma se reinicia la aplicación para aplicar los cambios.

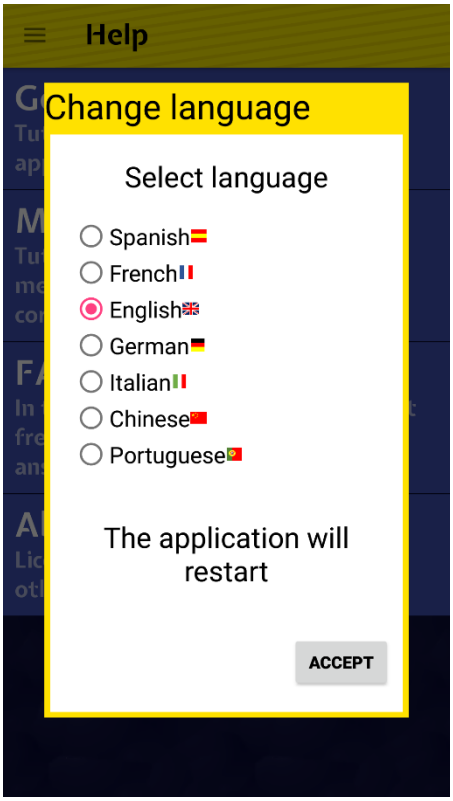


FIG. 36. DIALOG IDIOMA

Los elementos utilizados para la Fig.35 son: un NavigationDrawer a modo de contenedor, un archivo .xml a modo de header, y un archivo .xml a modo de menú como vimos en la estructura de la aplicación.

Los elementos utilizados para la Fig.36 son: TextView para los textos, RadioButtons para elegir una opción, e ImageView para las banderas.

5.4. PRUEBAS REALIZADAS

En este apartado se va a enumerar las pruebas realizadas para asegurar que la aplicación funcionaba en su totalidad, y las pruebas que se han llevado a cabo para poder formular la conclusión que se expondrá más adelante en este documento.

Se podrían explicar las pruebas realizadas en cada vista de la aplicación, pero sería alargar este documento innecesariamente. En la sección 5 se ha explicado en cada vista su funcionalidad, y se puede sobreentender que las pruebas realizadas son precisamente para comprobar que esas funcionalidades descritas están correctamente implementadas. Por ejemplo, registrar usuarios, activar usuarios, tomar medida y mostrar el resultado, cambiar datos de perfil, guardar históricos en BD, etc.

Entrando en pruebas relevantes, se ha probado la aplicación en varios dispositivos de distintos tamaños y distintos fabricantes para asegurar que la programación del interfaz de usuario de manera responsive funciona correctamente y todos los elementos se colocan de manera correcta. Por otro lado, se ha probado de manera adicional el sdk 23 para ver si todo funcionaba de igual manera. Para esto se ha hecho uso de un Xiaomi Mi Max con 6,44” y con versión de sdk 24, un Xiaomi Redmi 3S con 5” y con versión sdk 23, un Samsung Galaxy S4 con 5” y con versión sdk 23.

Otra prueba que realizada de cara a añadirle valor a la aplicación ha sido probar que es funcional para personas daltónicas. A través de una persona de mi entorno que lo padece, se han ido modificando las gamas de colores hasta conseguir que se desenvuelva sin dificultad.

La última prueba que mencionar antes de pasar a explicar las consideraciones sobre la toma de medidas de la respiración es una prueba de facilidad de uso realizada por los 5 usuarios de ayuda. Todos y cada uno se instalaron la aplicación y tuvieron que navegar sin ninguna indicación del desarrollador, pudiendo ver qué puntos cambiar para mejorarla.

En este punto se comienza con todas las pruebas y cambios que realizados para tratar de conseguir una medida precisa de la frecuencia respiratoria. Estas pruebas han sido las que han determinado la conclusión que se propondrá en otra sección, y las que más tiempo han supuesto debido a que se trata de la funcionalidad principal.

Se ha partido de una BD de datos de prueba de 6 usuarios: hombre 25 años, mujer 25 años, hombre 50 años, mujer 49 años, hombre 77 años y mujer 73 años. Se han realizado las mismas pruebas para todos ellos, de esta manera la conclusión tendrá una base sólida. A continuación, se muestran las pruebas para uno de los 6 usuarios comentados.

En primera instancia se iniciaron las pruebas restándole el valor del sensor de gravedad al sensor acelerómetro. Esto se planteó así porque de esta manera se obtiene el valor puro de aceleración. En reposo el acelerómetro coincide con la gravedad, así pues, con esta primera prueba en reposo se debería obtener una línea centrada en 0. Como vemos en la Fig.37. no se obtiene lo esperado, y esto se debe a que el acelerómetro no está bien calibrado.

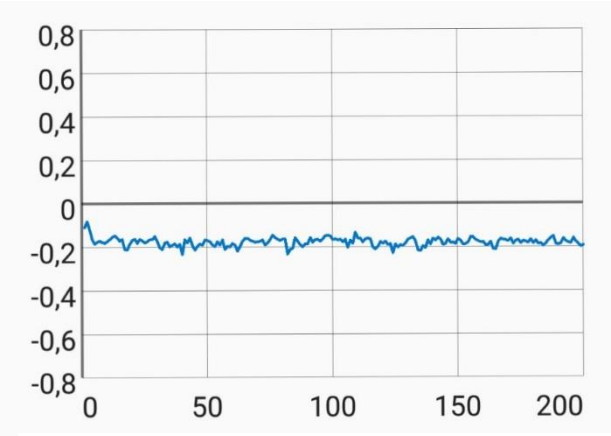


FIG. 37. ACELERÓMETRO NO CALIBRADO

Tras recalibrar los sensores del dispositivo de pruebas, observamos en la Fig.38. cómo se consigue obtener la línea centrada en cero que se comentaba antes.

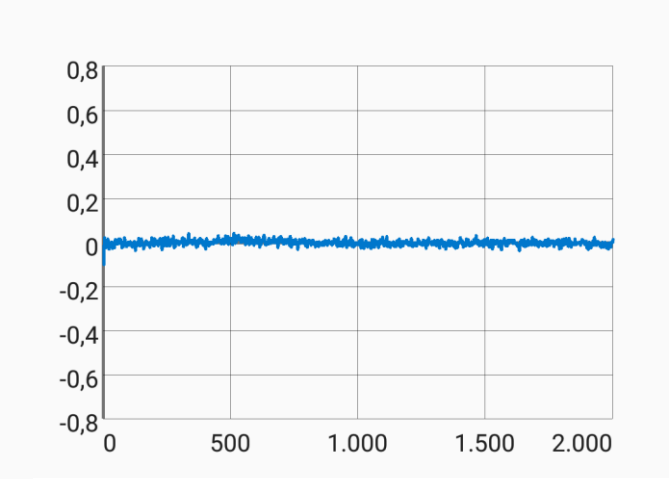


FIG. 38. ACELERÓMETRO CALIBRADO

A partir de la Fig.38. se puede observar que la señal está recibiendo mucho ruido A modo de orientación se simularon respiraciones profundas con la mano para obtener una gráfica "ideal". Ésta, debería tener una sinusoidal en cada respiración producida por la aceleración de la inspiración, desaceleración en el fin de la inspiración, aceleración en la expiración y desaceleración en el fin de la expiración. La Fig.39 representa dos respiraciones "perfectamente", dentro del ruido que hay.

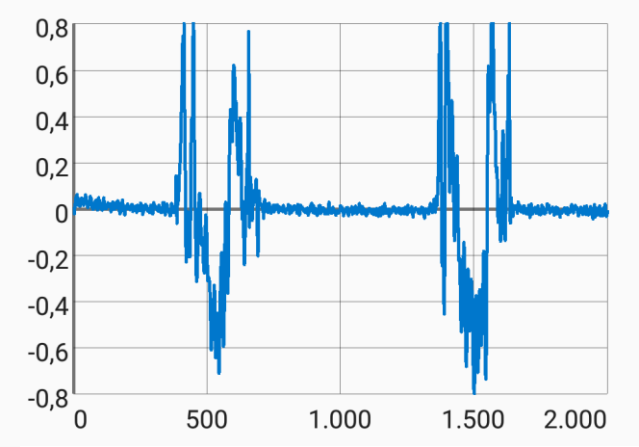


FIG. 39. GRÁFICA IDEAL

Con la Fig.39. en mente, se procede a tomar varias medidas con el dispositivo.

Los resultados obtenidos en este momento son gráficas totalmente ininteligibles como la Fig.40.

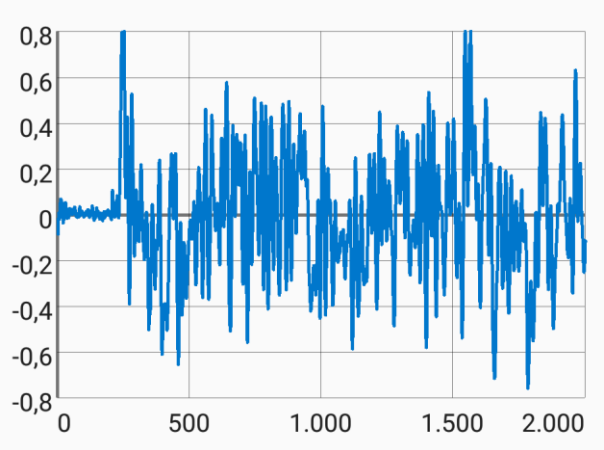


FIG. 40. MEDIDA INVÁLIDA

Para eliminar el ruido de la señar antes de intentar hacer operaciones sobre ella, se pasa la señal por un filtro paso bajo según la fórmula:

$$x[i] = \frac{x[i-1]+x[i]+x[i+1]}{3}$$

Con este método se consigue obtener una señal con mejor aspecto que todas las anteriores como se puede ver en la Fig.41. aunque sigue siendo muy complicado diferenciar las respiraciones.

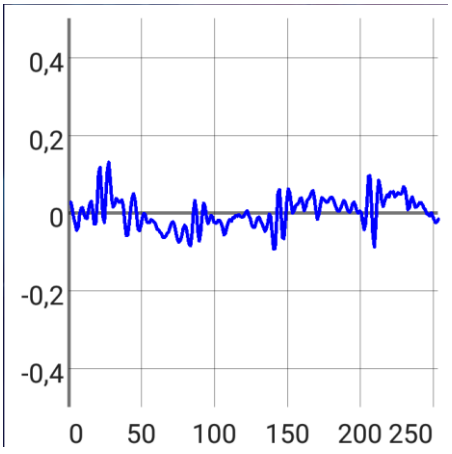


FIG. 41. GRÁFICA CON FILTRO

Para intentar afinar la gráfica se modifica la escala de los ejes, por si es problema de representación. Esto se puede observar en las gráficas anteriores que unas van de -0.4 a 0.4 y otras de -0.8 a 0.8. Además, se modifica la velocidad de los sensores. Mediante una variable se puede establecer a que frecuencia queremos que midan los sensores. Desde SENSOR_DELAY_FASTEST que es el más rápido, hasta SENSOR_DELAY_NORMAL que es el más lento. Finalmente, en cuanto al delay se decide establecerlo en SENSOR_DELAY_UI, que es la velocidad necesaria para interacciones de la interfaz de usuario. Estos cambios de delay se pueden ver en las gráficas anteriores en el eje x, unas van hasta 250, y otras hasta 2000.

Ninguna de estas configuraciones proporciona un resultado óptimo, así pues, se procede a testear estas configuraciones en otro dispositivo diferente por si el problema estuviera en los sensores del dispositivo de prueba. Pero tampoco se obtienen resultados satisfactorios.

Llegado a este punto, se decide realizar todas las pruebas anteriores utilizando únicamente el sensor de acelerómetro, es decir, sin restarle el valor de la gravedad como se planteó en primera instancia. Con las primeras pruebas tras este cambio se obtienen gráficas mucho más claras como la Fig. 42 y se empieza a intuir la forma sinusoidal en la Fig.43.

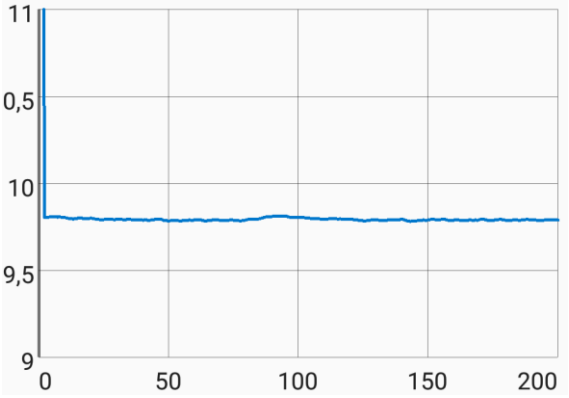


FIG. 43. ACELERÓMETRO SOLO

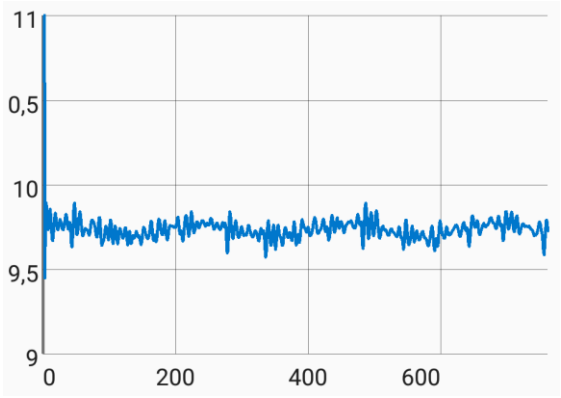


FIG. 42. SINUSOIDAL ACELERÓMETRO

La Fig.43. parece que tiene ruido, pero está filtrada al igual que la Fig.42.

Se plantea la posibilidad de que otros factores estén influyendo la señal. Haciendo pruebas más exhaustivas se llega a la conclusión que todos esos picos están producidos por vibraciones en el momento de las medidas y por las palpitaciones del cuerpo.

Para asegurar que efectivamente se trata de estas vibraciones y palpitaciones, durante la medida se procede a hacer un poco de presión sobre el móvil en dirección el cuerpo y se obtienen resultados como la Fig.44. Los resultados obtenidos sugieren que la aplicación no es una solución válida, ya que se trata de una situación controlada que los usuarios no controlan.

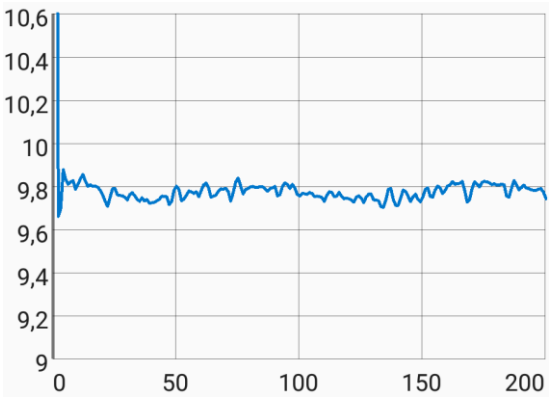


FIG. 44. GRÁFICA SIN VIBRACIONES

Pese a esto, se trata de continuar con esta solución y hacer operaciones sobre la señal para determinar el número de respiraciones. La solución tiene un inconveniente, y es que la presión ejercida sobre el móvil reduce la amplitud de la señal dificultando la posterior lectura de los valores para mostrar por pantalla el número de respiraciones realizadas.

Por último, se desarrolla el algoritmo que a partir de los valores me diga cuantas respiraciones se han realizados.

Para ello se han tenido que probar diferentes valores umbrales para considerar a un valor como un máximo. Esto es, se trata de medir las respiraciones buscando los máximos de la función. Este método tiene el problema que cómo la amplitud bajo estas circunstancias es mínima, cualquier pico que se acercara a los máximos era

considerado como una respiración más, dando como resultado muchas más de las debidas.

Dándole otro enfoque, y llegando a la proposición final antes de crear una conclusión, el algoritmo ha quedado de la siguiente manera: en primer lugar, se busca el máximo absoluto y el mínimo absoluto de la función en los n primeros valores, siendo n la ventana el tamaño de ventana de cada respiración. El tamaño de dicha ventana viene determinado por el peor caso según el standard de cada actividad realizada. Además, también se obtiene la posición del máximo absoluto. Con esta ventana y la posición del máximo absoluto, se procede a buscar el siguiente máximo. Se comienza a buscarlo n valores después del máximo absoluto, siendo n el tamaño de la ventana. Pero no se busca en la totalidad de la función, se busca el siguiente máximo en el siguiente tamaño de ventana * 2, y el valor de este nuevo máximo tiene que ser mayor o igual que el máximo absoluto – 0,05. Restando a la posición de este nuevo máximo, la posición del máximo absoluto del principio, y suponiendo que la respiración es simétrica en frecuencia, ya tengo cada cuántos valores se da una respiración.

En este momento, sabiendo que hay x valores en total en 15 segundos y sabiendo cada cuántos valores se da una respiración, se saca el número de respiraciones en 15 segundos y se multiplica por 4 para tener el valor en minutos.

Esta solución tiene el problema de basarse en que el dispositivo se encuentre en una superficie completamente plana, en caso contrario los valores no son válidos. Para asegurar que trabajamos en rangos correctos, en vez de suponer que la gráfica está centrada en 9.8, se calcula la media de la señal y se supone que está centrada en ese valor. De esta forma si el móvil no esta totalmente perpendicular a la superficie no nos afecta a la hora de medir.

Tras llegar a esta última solución, en alguna ocasión ha proporcionado un resultado correcto de medición. Pero teniendo en cuenta que cada persona respira con intensidad diferente, como se ha podido comprobar en los usuarios de prueba, y teniendo en cuenta que para llegar a este valor se ha creado una situación muy

controlada de medida, la funcionalidad de la aplicación queda en entredicho. Además, con este método de ventana se fuerza a buscar la respiración en bloques de ese tamaño, con lo cual si se respirara con más frecuencia se perderían respiraciones.

En vista a esto último, se realizó la última prueba eliminando el tamaño de ventana y buscando directamente el siguiente máximo a partir del primer máximo absoluto en lugar de un tamaño de ventana después. En este caso el resultado es peor, ya que cualquier pico que se acercara al máximo sin importar la distancia a éste se consideraba respiración y se desvirtuaba la medida.

6. POSIBLES MEJORAS Y PROYECCIONES DE FUTURO

En esta sección, se va a hablar de las posibles mejoras que se le podrían incluir al proyecto, y la utilidad que podría tener una aplicación de este estilo.

La primera mejora que cabe mencionar es la posibilidad de tomar la medida de la frecuencia respiratoria de pie. Tal y como se presenta la aplicación, y como se puede ver en el videotutorial general, solo se puede tomar la medida tumbado, así pues, una buena mejora sería poder tomarla de pie atándose el móvil al estómago o pecho con una cinta. Y como mejora final, se debería tratar de tomar la medida en movimiento.

Otra mejora de la aplicación sería crear "dos versiones", aunque realmente sea una, pero que haya distinción entre para uso particular, la cual se podría enfocar como una red social deportiva, y para uso médico, que serviría tal cual está la aplicación ahora. Para llevar a cabo la función que se propone de uso particular, lo primero sería cambiar la BD de SQLite a una externa de modo que se pueda intercambiar información entre usuarios. Tal cual está ahora, puesto que la BD es interna solo se tienen los datos en un dispositivo.

Introducir una función de tracking por GPS, podría ser interesante a modo de almacenar el recorrido que se ha realizado durante la actividad física, y así poderlo tener en cuenta para comparar medidas entre sí.

Por último, se propone la mejora de poder iniciar sesión en la aplicación mediante la huella dactilar. En las nuevas versiones de Android, y los nuevos dispositivos, esta funcionalidad es casi obligada.

En cuanto a las proyecciones de futuro, esta aplicación tiene mucho potencial en el ámbito de la salud. Sería un buen complemento utilizarla, por ejemplo, en la consulta del médico de cabecera. Esa primera auscultación por algún problema respiratorio no precisaría de material médico cualificado, tan solo se necesitaría un dispositivo móvil con la aplicación. Incluso ya podríamos ir desde casa con los resultados de la medida al médico. Esta proyección de la aplicación es una manera clara de ahorrar costes en la medicina básica.

También puede ser útil en los gimnasios, o para los entrenadores personales. Hoy en día el estar bien físicamente es algo que nos importa mucho, y por eso desde hace algún tiempo el negocio de los preparadores físicos está en auge. En ese caso, esta aplicación sería el complemento perfecto para esos entrenadores personales que pretendan dar un entrenamiento innovador. Podrían de manera complementaria observar los frutos de su preparación en sus clientes por la frecuencia respiratoria. Es una manera de profesionalizar sus servicios aún más. Y en los gimnasios algo parecido para usarlo a modo de reclamo para sus clientes presentándose como que utiliza métodos diferentes.

Las residencias de ancianos sería otro punto de explotación para esta aplicación. Las personas mayores que están internas en estos lugares precisan de un monitoreo prácticamente diario de sus constantes vitales. Así pues, de manera análoga a la proyección de la medicina primaria que se ha propuesto un poco antes en esta sección, una buena manera de ahorrar costes para este monitoreo de constantes vitales sería integrar en un dispositivo móvil todas esas medidas que necesitan. Siendo la frecuencia respiratoria una de ellas.

7. CONCLUSIONES

Para concluir este documento, este proyecto, y en vista de los resultados que se han expuesto en la sección 5, se puede decir sin riesgo de equivocación, que no es posible obtener una medida de la frecuencia respiratoria de forma precisa utilizando tan sólo el acelerómetro de nuestros smartphones. Para poder concluir lo contrario, el porcentaje de éxito en las medidas debería ser muy superior al obtenido.

Para la realización de las medidas, entran en juego demasiados factores: vibraciones en la superficie donde estés tumbado, palpitaciones del cuerpo, latidos del corazón, vibración de notificaciones del móvil, intensidad de la respiración del usuario, si la respiración se realiza en el pecho o en el estómago, etc. Todos estos factores y que como hemos visto en el apartado de pruebas se ha de hacer una medida muy controlada para obtener un valor bueno, se ha llegado a la conclusión descrita.

En el caso de que se hicieran algunas modificaciones en las especificaciones de la aplicación esta conclusión podría cambiar. Estas modificaciones de las que se hablan, puede que no estén muy lejanas. ¿Por qué se dice esto? En mi opinión con unos acelerómetros más potentes instalados en nuestros dispositivos podríamos mejorar la calidad de la señal obtenida y de esta manera aproximarnos más al resultado deseado. Del mismo modo, si se utilizaran wereables con sensores de calidad integrados también sería posible mejorar la calidad de la medición.

Puesto que lo interesante de esta aplicación es que funcione de manera automática no se ha llegado a un resultado satisfactorio. Si se necesita de una interacción del usuario por cada respiración, ¿qué utilidad tiene la aplicación? Sería innecesaria ya que el usuario podría contar las respiraciones por su cuenta.

Resumiendo, con la tecnología que tienen nuestros dispositivos instaladas actualmente no somos capaces de determinar con suficiente precisión cuantas respiraciones se hacen por minuto de manera automática. Recibimos valores provenientes del sensor acelerómetro que se desvirtúan por muchos factores externos. Sería necesario apoyarse en tecnología aparte o métodos que no se corresponden con las especificaciones que se han propuesto para esta aplicación

8. ABREVIATURAS Y TÉRMINOS

[1] **BD:** Base de Datos. Utilizada para almacenar datos de clientes, imágenes y preguntas frecuentes.

[2] **SO:** Sistema Operativo. Programa encargado de controlar los procesos y programas, en este caso, del dispositivo móvil. Hace referencia a Android.

[3] **Activity:** Cada una de las pantallas de las que consta la aplicación.

[4] **Responsive:** Manera de programar la interfaz de usuario para que se ajuste al tamaño de la pantalla del dispositivo.

[5] **Wereables:** También conocidos como periféricos. Se tratan de aparatos ajenos al dispositivo móvil, que se comunican con este para realizar funciones.

[6] **SDK:** Software Development Kit. Herramientas que permiten la programación de aplicaciones móviles.

[7] **Barra de navegación:** Hace referencia a la Barra superior que aparece en algunas aplicaciones donde se suele incorporar el nombre de la vista actual y los botones de navegación, como el menú, los ajustes, etc.

[8] **Tracking:** Literalmente se traduce como seguimiento. En este caso hace referencia al camino seguido durante una actividad a través del GPS.

[9] **GPS:** Global Positioning System. Es el Sistema que nos permite determinar nuestra posición física en el planeta.

[10] **MediaPlayer:** Se trata de una clase Java que nos permite controlar archivos de audio o video.

[11] **MediaController:** Hace referencia al cuadro que aparece durante la reproducción de un video o un audio, el cual contiene los controles (play, pause, fullscreen, rewind)

[12] **SurfaceView:** Se considera a un objeto Android que proporciona una superfice para realizar funciones sobre ella, en este caso se ha utilizado para contener los videotutoriales.

9. REFERENCIAS

[1] GSMA, *Página oficial de la GSM Association*. [En línea]. Disponible en: https://www.gsma.com/. Último acceso: 04/06/2018

[2] Población mundial a tiempo real. [En línea]. Disponible en: http://poblacion.population.city/world/. Último acceso: 04/06/2018

[3] Anónimo, Introducción - ¿Qué es Android?, *histinf*, 14 diciembre 2012. [En línea]. Disponible en: https://histinf.blogs.upv.es/2012/12/14/android/. Último acceso: 04/06/2018

[4] Digital55, iOS vs Android, tendencias y cuota de mercado, *Digital55*, 07/10/2017. [En línea]. Disponible en: https://www.digital55.com/ios-vs-android-tendencias-y-cuota-de-mercado/. Último acceso: 11/06/2018

[5] Statista, Number of apps available in leading app stores as of 1st quarter 2018, *Statista*, Mayo 2018. [En línea]. Disponible en: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. Último acceso: 11/06/2018

[6] Android Developers, Download Android Studio + Android SDK + ADB Plugin, *Android Developers*. [En línea]. Disponible en: https://developer.android.com/studio/index.html?hl=es-419. Último acceso: 20/11/2017

[7] Enrique Piñana, Aprende Android desde cero, 2012, *Aprende Android*. [En línea]. Disponible en: http://www.aprendeandroid.com/menu.htm. Último acceso: 03/12/2017

[8] Ramiro, Curso Android Studio, *Blogspot*. Disponible en: http://cursoandroidstudio.blogspot.com/. Último acceso: 03/12/2017

[9] Canal de Youtube Asesorías Maldonado, Mi primera aplicación en Android Studio, *Youtube*. [En línea]. Disponible en: https://www.youtube.com/user/bombinonet/videos. Último acceso: 03/12/2017

[10] Godfrey Nolan, Practical advice for building secure android databases in SQLite, *Informit*, 02/01/2015. [En línea]. Disponible en: http://www.informit.com/articles/article.aspx?p=2268753&seqNum=4. Último acceso: 16/01/2018

[11] Android Developers, Archivo Android.mk, *Android Developers*. [En línea]. Disponible en: https://developer.android.com/ndk/guides/android_mk. Último acceso: 16/01/2018

[12] Android Developers, Actividades, *Android Developers*. [En línea]. Disponible en: https://developer.android.com/guide/components/activities?hl=es-419. Último acceso: 04/12/2017

[13] Rebeca, 6 maneras de monetizar tu app, *Appnet*. [En línea]. Disponible en: https://www.tu-app.net/blog/monetizar-app/. Último acceso: 08/06/2018

[14] Sgoliver, Bases de Datos en Android (I): Primeros pasos, *sgoliver.net*, 31/01/2011. [En línea]. Disponible en: http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/. Último acceso: 13/01/2018

Sgoliver, Bases de Datos en Android (II): Insertar/Actualizar/Eliminar, *sgoliver.net*, 03/02/2011. [En línea]. Disponible en: http://www.sgoliver.net/blog/bases-de-datos-en-android-ii-insertaractualizareliminar/. Último acceso: 13/01/2018

Sgoliver, Bases de Datos en Android (III): Consultar/Recueprar registros, *sgoliver.net*, 07/02/2011. [En línea]. Disponible en: http://www.sgoliver.net/blog/bases-de-datos-en-android-iii-consultarrecuperar-registros/. Último acceso: 13/01/2018

[15] Zetetic, *Página oficial*. [En línea]. Disponible en: https://www.zetetic.net/sqlcipher/sqlcipher-for-android/. Último acceso: 15/01/2018

[16] EDMT Dev, Android Studio Tutorial – SQLite Cipher, *Youtube*, 29/10/2017. [En línea]. Disponible en: https://www.youtube.com/watch?v=mehBt5LJF84. Último acceso: 15/01/2018

[17] Suragch, Android Shared Preferences example, *StackOverFlow*, 21/11/2016, [En línea]. Disponible en: https://stackoverflow.com/questions/23024831/android-shared-preferences-example. Último acceso: 17/04/2018

[18] Designtown, Google fonts. [En línea]. Disponible en: https://fonts.google.com/specimen/Expletus+Sans. Último acceso: 21/05/2018

[19] Jamie Bearded-Hen, Android-Bootstrap, *Github*, 2015. [En línea]. Disponible en: https://github.com/Bearded-Hen/Android-Bootstrap/. Último acceso: 08/01/2018

[20] mcmurryjulie, Pulmones, *Pixabay*. [En línea]. Disponible en: https://pixabay.com/es/pulmones-pulm%C3%B3n-icono-respiraci%C3%B3n-2803208/. Último acceso: 01/06/208

[21] Miguel Campos, Cómo hacer notificaciones push en Android fácil, *openwebinars*, 06/06/2016. [En línea]. Disponible en: https://openwebinars.net/blog/como-hacer-notificaciones-push-en-android-facil/. Último acceso: 05/02/2018

[22] Indragni Soft Solutions, how to send email using Java Mail API in Android, *Youtube*, 25/09/2014. [En línea]. Disponible en: https://www.youtube.com/watch?v=UNPFWCNMJPU. Último acceso: 24/01/2018

[23] latincoder, Tomar foto de la camara y visualizar en ImageView, *Youtube*, 30/09/2013. [En línea]. Disponible en: https://www.youtube.com/watch?v=l2vBKhhk10s. Último acceso: 05/03/2018

[24] eyllanesc, Guardar imágenes en sqlite android, *StackOverFlow*, 01/07/2017. [En línea]. Disponible en: https://es.stackoverflow.com/questions/74332/guardar-imagenes-en-sqlite-android. Último acceso: 05/03/2018

[25] Andrea Ardións, Código para "enviar email" en Android studio, *Android Studio Faqs*, 09/01/2017. [En línea]. Disponible en: https://androidstudiofaqs.com/tutoriales/codigo-enviar-email-android-studio. Último acceso: 10/03/2018

[26] Daniel Ferrer Delgado, Cómo añadir eventos al calendario en android, *nosinmiubuntu*, 29/05/2013. [En línea]. Disponible en: https://www.nosinmiubuntu.com/como-anadir-eventos-al-calendario-en/. Último acceso: 13/03/2018

[27] Shane Conder, Android Fundamentos: Añadiendo Eventos al Calendario del Usuario, *envatotuts+*, 24/10/2011. [En línea]. Disponible en: https://code.tutsplus.com/es/tutorials/android-essentials-adding-events-to-the-users-calendar--mobile-8363. Último acceso: 13/03/2018

[28] Juan, Cómo pedir una fecha en Android usando DatePicker *PYM*. [En línea]. Disponible en: https://programacionymas.com/blog/como-pedir-fecha-android-usando-date-picker. Último acceso: 12/03/2018

[29] Fran García, 32. Hilos y AsyncTask. (Programación Android Studio tutorial español), *Youtube*, 06/11/2015. [En línea]. Disponible en: https://www.youtube.com/watch?v=cxUJJh29PBg Último acceso: 12/12/2017

[30] Jonas, GraphView open source library, *GraphView*. [En línea]. Disponible en: http://www.android-graphview.org/. Último acceso: 01/05/2018

[31] o7planning, Android MediaPlayer and VideoView Tutorial, *o7planning*. [En línea]. Disponible en: https://o7planning.org/en/10487/android-mediaplayer-and-videoview-tutorial. Último acceso: 20/05/2018

[32] Fernando Velázquez, Android-Video-Player-Example, *Github*, 30/10/2012. [En línea]. Disponible en: https://github.com/nandovelazquez/Android-Video-Player-Example/blob/master/src/com/demo/media/VideoStream.java. Último acceso: 09/04/2018

[33] Android Developers, MediaPlayer, *Android Developers*. [En línea]. Disponible en: https://developer.android.com/reference/android/media/MediaPlayer. Último acceso: 09/04/2018

[34] PVPDS, Navigation drawer with Activities in android studio (not fragments), *Youtube*, 24/05/2017. [En línea]. Disponible en: https://www.youtube.com/watch?v=M_4Oh2FeRYs. Último acceso: 29/03/2018

[35] Blodhgard, Android N change language programmatically, *StackOverFLow*, 24/08/2017. [En línea]. Disponible en:

https://stackoverflow.com/questions/39705739/android-n-change-language-programmatically/40849142#40849142. Último acceso: 02/04/2018